Biologically-inspired Network Memory System for Smarter Networking

Bassem Mahmoud Mohamed Ali Mokhtar

Dissertation submitted to the faculty of the Virginia Polytechnic Institute and State University in partial fulfillment of the requirements for the degree of

Doctor of Philosophy
In
Computer Engineering

Mohamed Y. Eltoweissy
Yiwei T. Hou
Luiz A. DaSilva
Sedki M. Riad
Ing R. Chen
Mohamed R. Rizk

January 21th, 2014

Blacksburg, Virginia

Keywords: Network Semantics, Biologically-inspired Networking, Network Intelligence, Network Management, Memory

Biologically-inspired Network Memory System for Smarter Networking

Bassem Mahmoud Mohamed Ali Mokhtar

Abstract

Current and emerging large-scale networks, for example the current Internet and the future Internet of Things, target supporting billions of networked entities to provide a wide variety of services and resources. Such complexity results in network-data from different sources with special characteristics, such as widely diverse users and services, multiple media (e.g., text, audio, video, etc.), high-dimensionality (i.e., large sets of attributes) and various dynamic concerns (e.g., time-sensitive data). With huge amounts of network data with such characteristics, there are significant challenges to a) recognize emergent and anomalous behavior in network traffic and b) make intelligent decisions for efficient and effective network operations.

Fortunately, numerous analyses of Internet traffic have demonstrated that network traffic data exhibit multi-dimensional patterns that can be learned in order to enable discovery of data semantics. We claim that extracting and managing network semantics from traffic patterns and building conceptual models to be accessed on-demand would help in mitigating the aforementioned challenges.

The current Internet, contemporary networking architectures and current tools for managing large network-data largely lack capabilities to 1) represent, manage and utilize the wealth of multi-dimensional traffic data patterns; 2) extract network semantics to support Internet intelligence through efficiently building conceptual models of Internet entities at different levels of granularity; and 3) predict future events (e.g., attacks) and behaviors (e.g., QoS of unfamiliar services) based on learned semantics. We depict the limited utilization of traffic semantics in networking operations as the "Internet Semantics Gap (ISG)".

We hypothesize that endowing the Internet and next generation networks with a "memory" system that provides data and semantics management would help resolve the ISG and enable "Internet Intelligence". We seek to enable networked entities, at runtime and on-demand, to systematically: 1) learn and retrieve network semantics at different levels of granularity related to various Internet elements (e.g., services, protocols, resources, etc.); and 2) utilize extracted semantics to improve network operations and

services in various aspects ranging from performance, to quality of service, to security and resilience.

In this dissertation, we propose a distributed network memory management system, termed NetMem, for Internet intelligence. NetMem design is inspired by the functionalities of human memory to efficiently store Internet data and extract and utilize traffic data semantics in matching and prediction processes, and building dynamic network-concept ontology (DNCO) at different levels of granularity. The DNCO provides dynamic behavior models for various Internet elements. Analogous to human memory functionalities, NetMem has a memory system structure comprising short-term memory (StM) and long-term memory (LtM). StM maintains highly dynamic network data or data semantics with lower levels of abstraction for short time, while LtM keeps for long time slower varying semantics with higher levels of abstraction. Maintained data in NetMem can be accessed and learned at runtime and on-demand.

From a system's perspective, NetMem can be viewed as an overlay network of distributed "memory" agents, called NMemAgents, located at multiple levels targeting different levels of data abstraction and scalable operation. Our main contributions are as follows:

- Biologically-inspired customizable application-agnostic distributed network memory management system with efficient processes for extracting and classifying high-level features and reasoning about rich semantics in order to resolve the ISG and target Internet intelligence.
- Systematic methodology using monolithic and hybrid intelligence techniques for efficiently managing data semantics and building runtime-accessible dynamic ontology of correlated concept classes related to various Internet elements and at different levels of abstraction and granularity that would facilitate:

    - Predicting future events and learning about new services;
    - Recognizing and detecting of normal/abnormal and dynamic/emergent behavior of various Internet elements;
    - Satisfying QoS requirements with better utilization of resources.

We have evaluated the NetMem's efficiency and effectiveness employing different semantics reasoning algorithms. We have evaluated NetMem operations over real Internet traffic data with and without using data dimensionality reduction techniques. We have demonstrated the scalability and efficiency of NetMem as a distributed multi-agent system using an analytical model. The effectiveness of NetMem has been evaluated through simulation using real offline data sets and also via the implementation of a small practical test-bed. Our results show the success of NetMem in learning and using data semantics for anomaly detection and enhancement of QoS satisfaction of running services.

Dedication

I dedicate my PhD dissertation work …

to my caring parents,
who are doing everything to motivate and encourage me to achieve success

to my lovely wife,
who stands always behind me and makes all of her best to support me

to my kids Abdelrahman and Youssuf,
who bear much time without me; and their smiles give me strong pushes to overcome challenges

Acknowledgements

First and foremost, all praises and thanks are due to Allah, the Almighty, for His graces and help throughout my life and research.

I am extremely grateful to prof. Mohamed Eltoweissy, my PhD dissertation advisor, who supported and helped me all the time to learn many things in different fields and for teaching me with care and patience how to be a good researcher. Prof. Eltoweissy taught me how to think outside the box and generate innovative ideas. His guidance directed me and my research work to better analysis, discussion and results. Prof. Eltoweissy treated me as a friend and a peer and this motivated me deeply to extract a lot of thoughts which enhanced my research work. The discussions I had with prof. Eltoweissy helped me tremendously to overcome many obstacles I faced in my PhD journey. I am looking forward to continue collaboration with prof. Eltoweissy in the future.

I would like to thank my committee members, prof. Thomas Hou, prof. Luiz DaSilva, prof. Sedki Riad, prof. Ing-Ray Chen and prof. Mohamed Rizk, for their valuable comments which guided me to enhance my dissertation work and improve the form and the quality of the dissertation.

I would like to thank prof. Ioannis Besieris, an emeritus professor in Virginia Tech, for his always support, help and hospitality. Prof. Besieris gave me great advices to enhance my research background and technical writing.

Special thanks and great respect I provide to prof. Sedki Riad for his dedicated work to establish the VT-MENA program. Prof. Riad did great efforts and gave many sacrifices to make the VT-MENA program achieve success. The VT-MENA program offered me a great chance to get my PhD form a prestigious university like Virginia Tech. Prof. Riad encouraged and helped me sincerely many times throughout my research work.

From the bottom of my heart, I would like to express my earnest gratitude to my parents and my brothers, Ahmed and Mostafa, for their support and encouragement all the time of my research. I am deeply indebted to my parents for everything they provided for me in my life to achieve success and to have a better life. Special thanks I provide to my brother, Mostafa Mokhtar, for his help and collaboration to improve my research work.

I am deeply beholden to my dear and caring wife for her love, encouragement, and patience throughout our life and my research study. She always gives me support to overcome obstacles and challenges to achieve success. I am deeply grateful to our children, Abdelrahman and Youssuf, for showing me their smiles which always push me to continue and enhance my work, especially in the cases of meeting great challenges.

At last but not least, I would like to thank my colleagues and all faculty and administrative staff in Virginia Tech and VT-MENA program whether in US or Egypt for their help and guidance to me.

Table of Contents

List of Figures

List of Tables

Chapter 1

1 INTRODUCTION

1.1 Motivation and Problem Statement

Due to semantically-oblivious networking operations, the current Internet cannot effectively or efficiently cope with the explosion in services with different requirements, number of users, resource heterogeneity, and widely varied user, application and system dynamics [1]. This leads to increasing complexity in Internet management and operations, thus multiplying challenges to achieve better security, performance and Quality of Service (QoS) satisfaction. The current Internet largely lacks capabilities to extract network-semantics to efficiently build behavioral models of Internet elements at different levels of granularity and to pervasively observe and inspect network dynamics. For example, a network host might know the role of TCP; however, it might not know the behavior of TCP in a mobile *ad hoc* network. We refer to the limited utilization of Internet traffic semantics in networking operations as the "Internet Semantic Gap (ISG)". Additionally, many evolutionary cross-layer networking enhancements and clean-slate architectures, see for example [2-5], did not consider capabilities for representing, managing, and utilizing the inherent multi-dimensional networking data patterns. Also, these architectures lack facilities to learn network-semantics and utilize them to dynamically allocate and predict "right-sized" services/resources on demand for example.

Both hardware and software solutions [6, 7] have been proposed for enhancing network intelligence (for example, cognitive networks) to better analyze and understand context (for example, traffic) and semantics of operating protocols (for example, routing protocols). Unfortunately, these solutions did not provide unified systematic means to represent, store, associate, and learn network-semantics at different levels of granularity (i.e., integrated cross-layer and cross-domain semantics) that can be used to extract information concerning various Internet elements (for example, attacks, resources, applications, etc.) under different operating systems in multiple operating contexts (for example., wired, wireless and mobile).

The current and future internetworks (for example, Internet of things (IoT) [8, 9]) support a massive number of Internet elements with extensive amounts of data. Fortunately, these data generally exhibit multi-dimensional patterns (for example, patterns with dimensions such as time, space, and users) that can be learned in order to extract network-semantics [10]. These semantics can help in learning normal and anomalous behavior of the different networking elements (for example, services, protocols, etc.) in the Internet, and in building behavior models for those elements

accordingly. Recognizing and maintaining semantics as accessible concepts and behavior models related to various Internet elements will aid in possessing intelligence thus helping elements in predicting future events (for example, QoS degradation and attacks) that might occur and affect performance of networking operations. Furthermore, learning behavior of those elements will better support self-* properties such as awareness with unfamiliar services and also advance reasoning about their behavior. For instance, a router can classify a new running service in a network as a specific type of TCP-based file transfer service when it finds similarity between behavior of the new service and that of an already known service.

For efficient storage, access and processing of big network-data generated from large scale networks, the following network-data characteristics should be addressed:

a) The massive amount of network-data with continual growth;
b) The high and multi-dimensionality of network-data (i.e., huge number of network attributes related to various network concerns) which require tremendous storage space;
c) Dynamicity: ever-changing data volume, data transfer rate and data features with time and space;
d) Variety in information (e.g., structured data as images and unstructured data as text); and
e) Complexity according to the variety in the used data representation models and language.

Due to the above mentioned network-data characteristics and limitations in entity capabilities and resources, entities may not be able to manage and analyze autonomously big data, learn patterns and extract semantics. There are other external factors which limit the capabilities of entities, such as the narrow operation scope per entity to handle high-dimensional data. Unfortunately, the current Internet and the proposed network architectures, see for example [2, 5] in the contemporary literature, for the most part, do not resolve the ISG. Contemporary tools and solutions as presented in [11, 12] manage large real Internet data sets. Multiple solutions, such as hadoop [13], have been investigated to support efficient storage, processing and analysis for big data of distributed applications. However, they are limited in their ability to identify dynamic behavior aspects, and thus they constrain our understanding of actors and activities in the network. Moreover, the storage of traffic data for mining and analysis would be prohibitive given the extreme volume of data and the timeliness needed in decision making.

The lack of efficient methodology and capabilities for analyzing and learning patterns of high- and multi-dimensional big network-data and reasoning about network-semantics presents challenges including but not limited to the following:

- Recognizing emergent and abnormal behavior of various Internet elements;
- Making effective decisions for efficient network operations;

- Ensuring availability of resources on-demand; and
- Efficient utilization of networked entities' capabilities to store, access and process data and extract valuable network-semantics.

1.2 Scenario

In this subsection, we provide a simple working example in order to motivate and illustrate the need for our proposed Network Memory Management System, or NetMem, its design and functionality, and to show the limitations of the current Internet [1] and contemporary networking architectures. We assume an Internet-based scenario, as illustrated in Figure 1.1, which comprises heterogeneous Internet elements and networking environments with large scale of generated high-dimensional Internet data. Communication channels are established via the interconnecting infrastructure enabling data packets transfer among different environments. Flows of legitimate service (e.g., web browsing and database files uploading and reading) can be established between Internet-enabled entities and FTP and web servers. Some non-legitimate TCP-based service flows are initiated from malicious entities (Attackers A and B). Attackers generate anomalous TCP- and UDP-based service flows with enormous amounts of packets for suppressing legitimate services and consuming storage resources of FTP and web servers and routers. Consequently, the generated anomalous flows from attackers impede meeting QoS requirements of running legitimate services. Intermediate routers have no capability to learn autonomously, at runtime and on-demand, data patterns and semantics concerning the normal and abnormal behavior of file transfer services in networking environments (e.g., normal/abnormal behavior of a TCP-based file transfer service in IP networks). In addition to network entity resource limitations, characteristics of Internet data, such as massive volume and high-dimensionality, prevent network

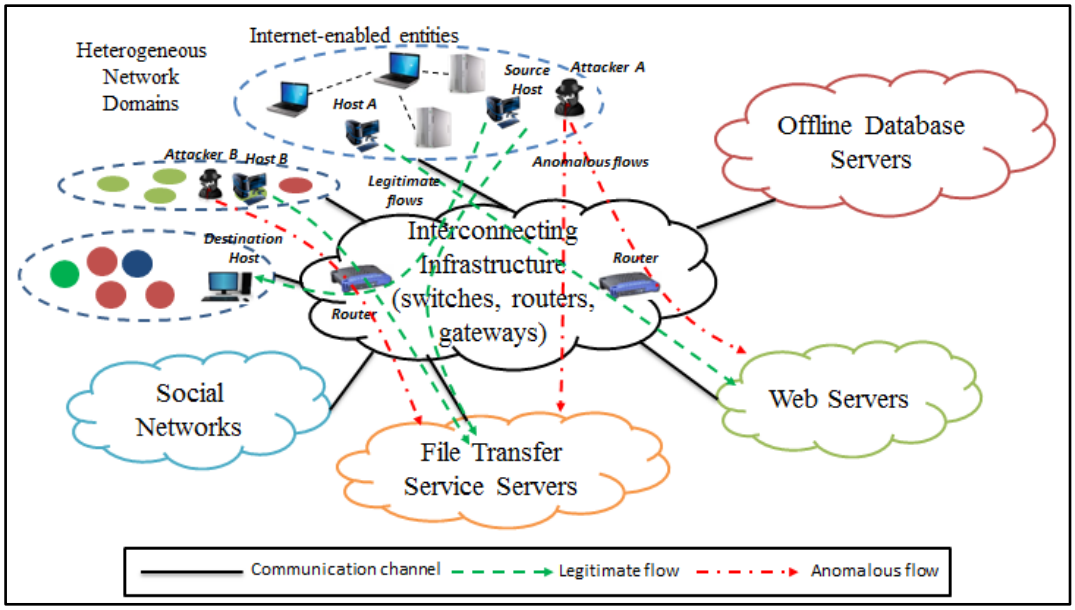


Figure 1.1 DoS network attack scenario

entities from learning data patterns and reasoning about semantics. In the sequel, we provide two scenarios based on this example. Section 1.4 will show the role of NetMem in providing capabilities for learning patterns and behavior concepts of unfamiliar services and predicting future events.

1.2.1 Learning QoS requirements of interesting and unfamiliar services

In this scenario, we demonstrate the need for NetMem to help network entities in learning and meeting QoS demands of their services and services they support in order to preserve established communication channels among entities.

Scenario description

As illustrated in Figure 1.1, communication channels are established to transfer legitimate service data among network entities. We have different operation cases as described in the following paragraphs.

Case (1):

The ***source host*** sends the ***destination host*** asking for running a file-transfer service. The ***source host*** clarifies the target of the service "reliable file transfer", the used communication protocol "TCP", communication port numbers, the default packet size, etc., . The ***destination host*** has no previous knowledge about QoS requirements of that service. As a result, the connection might not be established and, if it is done, the QoS demands of the required service might not be met.

Case (2):

Host A and ***Host B*** are interested in two services offered by a web server and a FTP server, respectively. ***Host A*** and ***Host B*** have previous knowledge about these services and QoS requirements, such as the maximum allowed latency. However, they need to learn updates about QoS demands of their services in order to aid in the optimization of QoS and the utilization of their resources. Unfortunately, ***Host A*** and ***Host B*** have no capabilities to learn autonomously the required knowledge. Additionally, intermediate routers have no experience about QoS requirements and communication concerns (e.g., recommended the routing protocol) of supported services for ***Host A*** and ***Host B***. However, routers have no ability to get their required information. Consequently, the QoS of running services might be deteriorated due to the limitations of

1- Current Internet:
- Semantically oblivious protocol operations
- No self- knowledge, analysis or learning capabilities
- Best effort QoS

2- Contemporary networking architectures:
- The incapability to provide facilities for learning network-semantics and represent those semantics at different levels of granularity showing the different network concerns and functional, behavioral and structural (FBS) aspects.

The results: a) connection failure amongst the source and the destination hosts and b) deterioration in the QoS of running services between hosts A and B and web and FTP servers. Figure 1.2 depicts the impact of having no capabilities for learning network-semantics and presenting those semantics as ontology of network concepts that can be learned by network entities at run-time and on-demand.

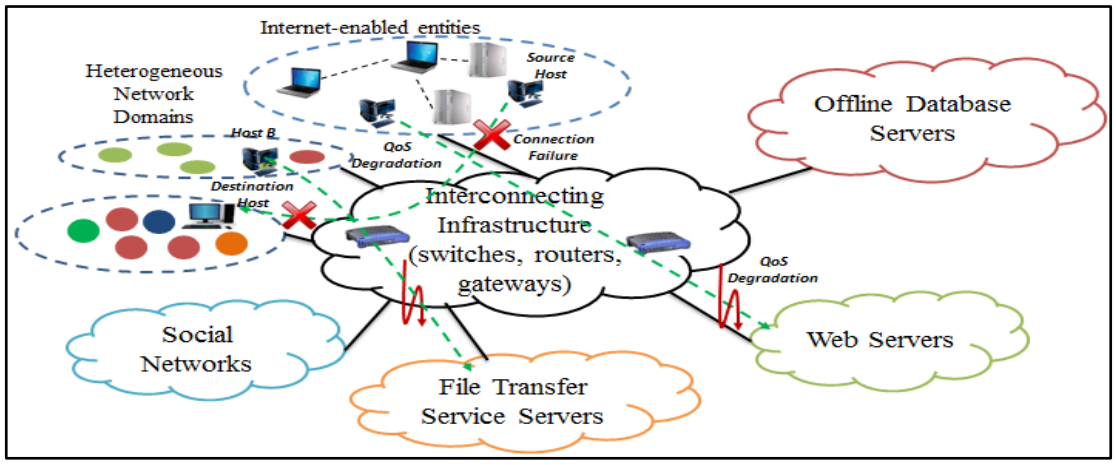


Figure 1.2 QoS deterioration and connection failure due to unawareness of QoS demands

1.2.2 Learning normal/abnormal behavior of service traffic flows

In this scenario, we demonstrate the need for NetMem to help network entities in learning the normal/abnormal behavior of services they support in order to preserve QoS requirements of legitimate services and suppress the non-legitimate ones.

Scenario description

Communication channels of legitimate running services are established to transfer data among network entities as depicted in Figure 1.1. One legitimate service flow is established between the source host and the FTP server. There are two non-legitimate services running by two attackers (attacker A and attacker B). ***Attacker A*** and ***attacker B*** generate anomalous TCP- and UDP-based service flows, respectively. The ***source host*** wants to communicate with: a) a FTP server to execute a file transfer service (e.g., uploading a database file), and b) a host to share a file with it. The generated attacks from the other two hosts (***attackers A and B***) deteriorate the QoS of legitimate running services. Intermediate routers have no previous knowledge about the normal/abnormal behavior of running services. Consequently, the ***source host*** is not able to accomplish its required job.

In the current Internet and contemporary networking architectures, there are limited capabilities for learning traffic patterns to reason about semantics that can be used to build behavior models classifying the normal/abnormal behavior of various Internet elements within specific operation contexts (e.g., normal/abnormal behavior of routing protocols and file transfer services in wired networks). Built behavior models can clarify classes of abnormal behaviors, such as defining the type of attacks and showing the FBS

aspects of those attacks. Due to the above mentioned limitations, unknown abnormal services deteriorate the QoS of the legitimate service and consequently the source host fails in having its required service fulfilled, as shown in Figure 1.3.

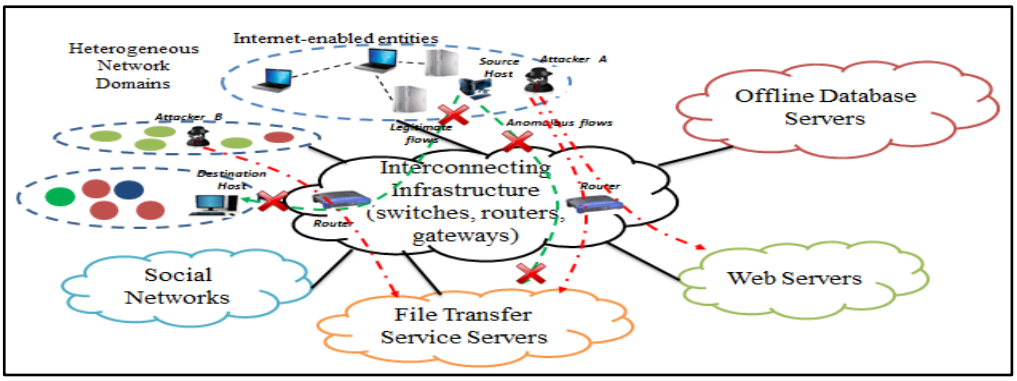


Figure 1.3 Interruption of services due to DoS attack

In the following subsection, we will discuss our research approach to mitigate the challenges illustrated in the above scenarios.

1.3 Research Approach

As stated earlier, there is a need to endow Internet operations and running services and applications with intelligence. In the literature, Internet (or network) intelligence (InetIntel) is defined as the capability of Internet elements to understand network-semantics in order to be able to make effective decisions and use resources efficiently [14]. InetIntel has to support Internet elements with the capability for learning normal and dynamic/emergent behavior of various elements and, in turn, building dynamic behavior models of those elements. InetIntel can be achieved via employing intelligence techniques to efficiently reason about semantics from large amount of Internet traffic raw data and provide runtime accessible valuable information at different levels of granularity. InetIntel should be achieved in a way that will not negatively impact Internet robustness or scalability.

To learn and efficiently extract network-semantics, network-data exhibiting patterns should be maintained; and their patterns have to be learned in a systematic way. Network-data can be captured from data traffic via various sources, such as a) Internet traffic data packets via sensory systems; b) offline databases; and iii) management information bases at network entities, like hosts.

The Internet has a tremendous and an ever-growing scale. It is noisy and dynamic with dissimilar communicating networks and heterogeneous entities, running diverse services and resources. Performance of intelligence techniques might be negatively affected by generated and transmitted Internet data with characteristics stated in Section 1.1. There is a need to homogenize data from various sources and normalize their ranges so that their patterns can be learned in a unified and systematic way. For these reasons, we conjecture that current Internet and future networks (e.g., IoT) should be endowed with a “memory” management system to efficiently store and manage multi-

dimensional data patterns in order to reason about semantics related to various Internet elements. These semantics can help in

a) Identifying and predicting network dynamics and emergent behavior;

b) Learning normal and anomalous behavior of the different Internet entities (e.g., services, protocols, etc.); and

c) Building behavior models for those entities to aid in predicting future events (e.g., attacks) that might occur and affect the performance of networking operations.

As an outcome, our solution will strengthen networking entities self-* properties, such as awareness of unfamiliar services and reasoning about their behavior. Also, it will enable networking entities, via systematic ways, to access and retrieve maintained semantics at run-time and on-demand.

NetMem Goal and Overview:

We propose NetMem, a shared distributed "data and semantics" memory management system that can be realized using distributed inter-connected autonomous intelligent agents to learn network-data patterns and reason about network-semantics in order to resolve the ISG. Figure 1.4 illustrates an abstract view of NetMem. NetMem targets achieving Internet intelligence through smarter semantic-driven operations in current highly dynamic complex networks as well as emerging networking environments. NetMem can be attached to already shared, existing and interconnected networking services, e.g., DNS services or network management services, in the current Internet. NetMem provides capabilities for Internet elements to discover/learn/retrieve, at real-time and on-demand, two types of data: 1) detailed data with low levels of abstraction (e.g., service data profile attributes, such as service type, packet size, routing protocol, etc.); and 2) highly abstracted data semantics and concept classes related to various network concerns (e.g., the normal/abnormal behavior of TCP and UDP protocols in wireless environments).

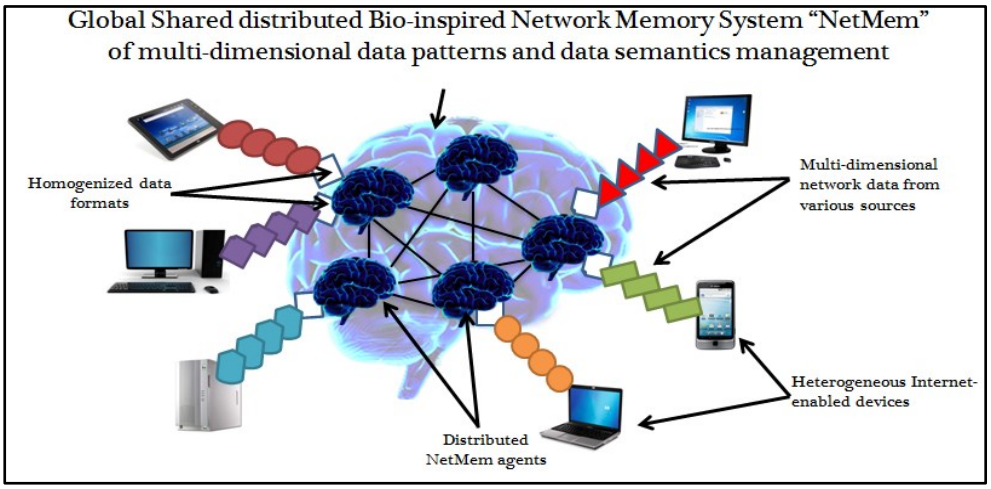


Figure 1.4 Abstract view of the NetMem system

In the complex large-scale Internet, huge amounts of multi-dimensional network-data are continuously generated from various sources [10]. Due to network-data characteristics, NetMem adopts Data Virtualization (DV) techniques [23] and dimensionality reduction algorithms to efficiently represent and manage diverse raw network-data. NetMem utilizes DV techniques with a defined data model to store uniformly any sort of data that exhibit patterns as profiles of attribute-value pairs. Additionally, we utilize appropriate dimensionality reduction algorithms, specifically Locality Sensitive Hashing (LSH) [15], to reduce the amount of data stored without significant loss of information.

Since networking traffic flows exhibit multi-dimensional data patterns, semantics can be extracted from learning network-data patterns, and they can be stored and retrieved using functionalities akin to human-memory functionalities [16]. NetMem mimics the human memory functionalities of deriving/matching semantics based on learning patterns and its capabilities for the associative storage and retrieval. NetMem maintains recognized semantics as correlated groups of network concepts at different levels of granularity.

NetMem forms Dynamic Network-Concept Ontology (DNCO), which is updated over time covering various and correlated classes of concepts. Stored network concept classes at DNCO are related to different Internet elements with different levels of abstraction and at different levels of granularity. Each maintained concept will be classified according to three network concerns: Application, Communication and Resource (ACR), as defined in [5], showing its functional, behavioral and structural aspects. NetMem would store any sort of data exhibiting patterns which can be learned to extract semantics. NetMem uses cloud-like data storage techniques [17] to store network-data and extracted semantics and the DNCO. Internet elements can learn semantics at runtime and on-demand, as well as related concepts in DNCO, for enhancing various networking operations and services, such as anomaly detection, resource optimization and QoS satisfaction.

For learning multi-dimensional data patterns and extracting network-semantics related to various network concerns, NetMem executes semantic management processes using: a) machine learning algorithms for learning groups of data attributes; b) feature extraction and classification algorithms for recognizing classified data features; and c) intelligence techniques-based reasoning models for identifying semantics based on extracted sets of data features.

The NetMem system consists of groups of communicating and interacting NetMem agents, or NMemAgents, at different NetMem system levels. Each NetMem level presents an abstraction level for maintained data/semantics defined by the content of the storage memory in each comprised NMemAgent. Each level has a higher level of abstraction than the level below it. NMemAgents at lower NetMem levels will maintain raw data and semantics with lower levels of abstraction and agents at higher levels will maintain data semantics with higher levels of abstraction.

Additionally, the storage duration of concepts maintained in upper NMemAgents is longer than the one for concepts in lower NMemAgents. So, the behavior of storage memory in agents at lower NetMem levels resembles StM, while it is closer to LtM in agents at upper levels. The hierarchy of concepts extracted by NMemAgents at different NetMem levels and with various abstraction levels enables building DNCO related to various network concerns. Internet elements (e.g., hosts) in various networking environments can access and request extracted information in the formed DNCO via communication with NMemAgents at the lowest level.

Hypotheses:

1. Networks generate huge amounts of data originating from heterogeneous sources. These data exhibit multi-dimensional patterns. Endowing networks with "memory" that has functionalities similar to human memory with short-term raw data storage and long-term semantics storage would provide Internet elements with capabilities, at run-time and on-demand, for
   - Learning and retrieving network-semantics at different level of granularity related to various elements; and
   - Utilizing extracted semantics to improve network operations and services in various aspects, ranging from performance, to QoS, to security and resilience.
2. Reasoning about network-semantics with high accuracy via learning patterns of low/high data volume with reduced data dimensionality would help in
   - Building conceptual models and ontology which can aid at real-time, for instance, in learning normal/abnormal behavior of various Internet elements and enhancing QoS of existing networking tools such as intrusion detection systems.
3. The complexity (space and time) of implemented algorithms for learning patterns and semantics reasoning of large scale multi-dimensional network-data can be reduced by applying data dimensionality reduction techniques while achieving accurate semantics reasoning processes.
4. Customizable application-agnostic network memory management system with runtime on-demand accessible ontology of associated network-semantics related to various network concerns would help Internet elements enhance/support:
   a. Self- and situational-awareness capabilities
   b. QoS of interesting services
   c. Predictive operation
   d. Anomaly detection
   e. Existing networking tools such as intrusion detection systems.
   f. Awareness of QoS demands of unfamiliar services

Underlying assumptions

In this dissertation, we have made the following assumptions:

- Internet traffic exhibits multi-dimensional patterns that can be learned based on various dimensions (space, time, Internet element, network concern, etc.) to reason about semantics; and
- Raw network-data can be classified according to three main network concerns that are defined as application, communication, and resource concern.

Analogy of NetMem with the functionalities of the human memory

We investigated the "human memory" model [16] and guided by the fact that networking operations exhibit multi-dimensional data patterns, the NetMem system is inspired by the functionalities of the "human memory". Human memory is capable of autonomously collecting huge amounts of data with different levels of detail through the five senses, learns their patterns and derives associative semantics accordingly. There are unified semantics representations and a scalable structured way for yielding associative semantics storage. Moreover, semantics in the human memory are accumulated as sequences, which are updated and can be associatively accessed and retrieved. Based on maintained semantics, humans can predict future events, learn things, and recognize new ones by matching their estimated semantics with those that are already registered. Our claim is that the functionalities of human memory [16] are suited to designing NetMem because of analogy in having patterns exhibited by raw data, processes of deriving/matching semantics based on learning patterns and capabilities which are offered in human memory for the associative retrieval and scalable storage. Numerous analyses of Internet traffic have demonstrated that network traffic data exhibit multi-dimensional patterns [10]. Those data patterns have different factors based on which patterns can be analyzed and learned. For example, data patterns can be studied on the basis of space-based features (e.g., IPs and server domains), time-based features (e.g., certain time slots per day), user and service (i.e., Internet element) characteristics, etc. These features and characteristics relate to data attributes captured from raw data. High level network-data features and data semantics can be extracted and learned by studying the patterns of raw data. Utilizing network-semantics will aid in forming conceptual and behavior models of networked entities and Internet elements. Learned semantics and models can be adopted to build dynamic and accessible ontology of network concepts. Network concepts ontology would help in mitigating challenges emerging due to current Internet and future networks and related generated big network-data.

Design Principles

- *Separation of concerns: application, communication, resource, and knowledge*: Internet elements and their relevant data are classified as Application, Communication and Resource (ACR) concerns and the modeled data which represent knowledge about Internet elements forms the knowledge concern.

- *Composable and cooperative building components learning patterns and semantics reasoning processes:* Self-managing memory system architecture relies on building blocks which comprise components that can be assembled to interact and perform tasks of capturing raw data, learning data patterns, reasoning about semantics, and handling requests concerning required data/semantics by network entities.
- *Associative concept access emphasizing multi-dimensional patterns:* The ability to discover and retrieve a concept associated with other concepts related to various network concerns and based on knowing part of that concept which might relate to different aspects: functional, behavioral and structural aspects.
- *Federated distributed memory management:* The ability to manage data retained in a distributed memory which comprises components where these components can interact and control tasks in memory.
- *Unified data model accommodating heterogeneous sources of high-dimensional large scale data:* The usage of a data model that accommodates efficient representation and storage for data generated from various sources with huge amounts and high-dimensionality.
- *Abstraction of network-semantics into functional, behavioral, and structural aspects:* The capability to show network-semantics through three different dimensions depending on the FBS engineering framework.
- *Representation of network-semantics as dynamic ontology of network concepts at different levels of abstraction:* The ability to form dynamic ontology of network concepts based on extracted network-semantics.

NetMem Design Methodology

Our methodology for designing NetMem is realized as shown in Figure 1.5 through the following concepts and operations:

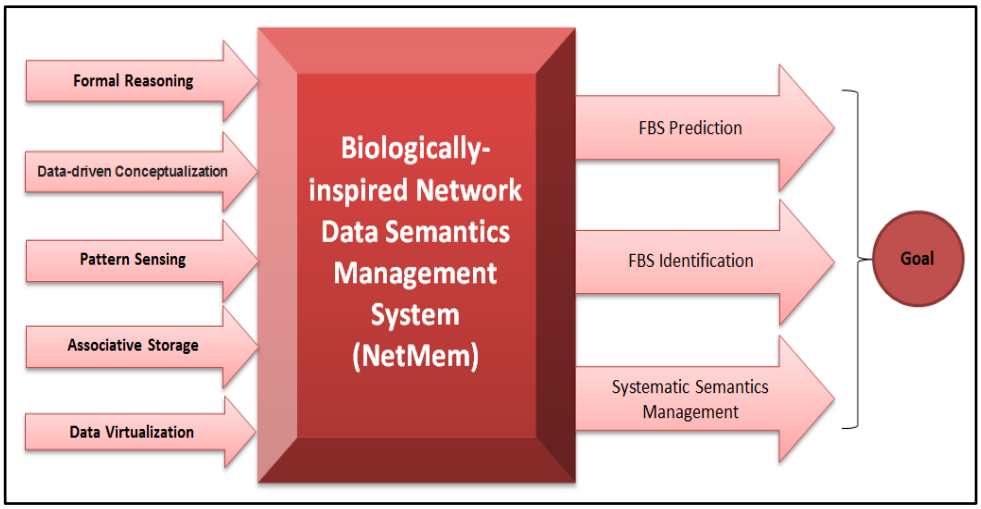


Figure 1.5 Data Semantics Management Methodology

a. <u>System Output (the main features to achieve our goal)</u>

1- *FBS Prediction:* The capability of a system to conjecture semantic data at different levels of abstraction which are functional, behavioral and structural aspects of various Internet elements (e.g., nodes, services, applications, protocols, etc.) in case of normal operation modes or emergent behavior due to adaptation and evolution capabilities for elements. This is performed through learning their patterns and based on maintained concepts and related associative conceptual patterns, and by applying implemented reasoning and concept matching algorithms. For instance, service A has patterns and by learning them and extracting features, associated concepts (i.e., data semantics) can be derived using concept models. These semantics would be matched using appropriate algorithms on different data aspects with already maintained semantics to discover new concepts related to that service:
    a. *Feature benefit:* Predict semantics at different levels of granularity for various Internet elements to understand their roles, requirements, behavior, and structure.
    b. *Feature dependence:* 1) The implemented reasoning models "or concept derivation system" which includes concept prediction algorithms, 2) semantic data structured warehouse and 3) associative retrieval mechanisms.

2- *FBS Identification:* The ability of the system to learn patterns of traffic related to communicating entities and running services and applications in complex systems to identify data semantics at different levels of abstraction (i.e., functional, behavioral, and structural data) in case of normal operation modes or emergent behavior due to changes in internal/external contexts. FBS identification would depend on maintained concepts and via applying concept matching algorithms.
    a. *Feature benefit:* Identify semantic data of normal operation for various Internet elements and, also, their emergent behavior to determine most probable function, behavior, and structure based on comparison with existing and maintained semantics in the system's semantics warehouse. In case of emergent behavior, the system enables understanding of sudden/abnormal changes in complex networking environments which might lead to networks instability and deterioration in QoS of running services and applications.
    b. *Feature dependence:* 1) the implemented reasoning models "or concept derivation system" which includes concept matching algorithms 2) semantic data structured warehouse and its associative access ability.

3- *Systematic Semantics Management:* Semantics are maintained and organized in a structured way forming an ontology of concepts, using conceptual models and definition language besides showing different levels of details as functional, behavioral and structural aspects. Semantics are stored as associative groups of concepts which reveal relationships among learned data semantics. In addition, semantics and any level of their abstracted data can be retrieved through an auto-associative access capability for performing concept matching processes at different levels of granularity.
    a. *Feature benefit:* Associative and organized network-semantics that facilities the discovery of concepts and their interdependence among other concepts. This will lead to fine tuning matching result when it is intended to retrieve most probable semantics data of an Internet element.
    b. *Feature dependence:* 1) Semantics structured warehouse and its associative access ability, 2) semantics storage and retrieval mechanisms and 3) concept models and definition languages.

b. <u>System input (the main fundamental concepts to achieve our system output):</u>

1- Formal Reasoning: This concept states a well-founded artificial intelligence functionality based on integrated probabilistic and statistical reasoning models to perform semantics reasoning and matching. These models will be used to extract semantics from learned data patterns and already known semantics. Analogy to the human memory [16], constructed chains of neurons at different cortex locations (e.g., vocal and visual cortex areas) of the human brain result in the formation of high level neurons. Capturing data, which infer to known information in the brain, makes sequences of neurons in various related cortex areas form.
    a. Concept benefit: it provides intelligence property to the system enabling it to discover semantic meanings and remove ambiguity in data in the contextual knowledge (i.e., resolve the semantics gap).
    b. Concept dependence (from our system perspective): 1) algorithms for features extraction and classification, 2) concept prediction algorithms, 3) concept similarity matching algorithms.

2- Data-driven Conceptualization: It is the operation for learning and deriving concepts (i.e., Semantics) based on extracted and classified features of learned data patterns. Hence, data and their learned patterns lead to concept extraction. This is done in the human memory [16] whereby patterns of data captured from various senses are learned by knowing represented sequences of associated neurons in various cortex areas.

a. *Concept benefit:* It generates semantics (high level of abstraction) from raw data (low level of abstraction) which are features of learned patterns.
b. *Concept dependence (from our system perspective):* 1) Concept prediction algorithms which are used to extract concepts; 2) concept models and definition language for unified representation of concepts.

3- *Pattern Sensing:* It provides the approach for learning patterns related to data of various Internet elements, such as services, applications and protocols. Recognizing patterns depends on their extracted and classified features, which are related to three networking concerns, namely; application, communication and resource concerns. In the human memory [16], sequences of neurons in certain cortex areas (e.g. visual and vocal areas) with certain connection pattern lead to identifying characteristics of that pattern. For instance, hearing and seeing a cat leads to a certain neuron pattern in our brain based on learned concepts.
   a. *Concept benefit:* It aids in understanding patterns which relate to raw data to learn and extract concepts.
   b. *Concept dependence (from our system perspective):* 1) Collected data from sensory system, 2) data grouping and classification algorithms, 3) data similarity matching algorithms, 4) learning algorithms and 5) feature extraction and classification algorithms.

4- *Associative Storage:* This concept shows that storage would be enabled with capabilities of identifying storage locations by their contents or part of the contents and having hierarchical storage structure by showing interdependence in different storage locations based on some criteria, e.g., imply or dependence. This matches operations of short-term and long-term memories in the human brain [16]. Low or high levels of neurons fire in different regions in the cortex when they capture data, which indicate to different types of information. These groups of neurons are connected as sequences. They are referring to lots of detailed/abstracted data.
   a. *Concept benefit:* It enables auto-associative data access to aid in having efficient similarity data/concept matching processes
   b. *Concept dependence (from our system perspective):* 1) Data models; 2) concept models; 3) data structured big relational tables

5- *Data Virtualization (DV):* DV is a technical process for abstracting/ federating/presenting massive data from heterogeneous sources to one logical place without physical data movement. DV provides for our system built-in tools for data homogenization and, also, data models for unifying data representation. DV uses algorithms for data grouping, classification, and similarity matching. This leads us to functionalities of the sensory system in the human brain [16].

That system collects huge amount of data from five senses and sends it to the brain via nerve signals, i.e., unified data representation.

a. *Concept benefit:* It provides unified representation for raw data originated from diverse sources.
b. *Concept dependence (from our system perspective):* 1) Collected data from sensory system; 2) data grouping and classification algorithms; 3) data similarity matching algorithms; 4) data models; 5) structured raw data warehouse in the short-term memory.

c. <u>System Components (our methodology to implement input and achieve output):</u>

- NetMem: a shared distributed biologically-inspired network memory management system of network-semantics mimicking the functionalities of the human memory. Our system consists of groups of agents, we call NMemAgents, at different levels based on abstraction level of data or semantics each agent maintains. Each NMemAgent comprises a set of composable and cooperating building blocks forming processes of i) data feed-forward for patterns learning and semantics reasoning, and ii) semantics feedback for matching and prediction tasks. NMemAgent main components are:
  1- Data collection and acquisition component mimicking the sensory memory system in the human brain
     - Sensory system to gather raw data from internal systems (e.g., about internal context in entities as running resources and operating protocols using their management information bases (MIBs) [18]) and from external contexts (e.g., data traffics through various communication channels).
     - Data virtualization techniques for homogenizing huge amounts of high-dimensional data and representing data uniformly.
     - Data models using the function-behavior-structure (FBS) engineering framework [19, 20] for abstracting network-data to functional-behavioral-structural attributes.
     - Data dimensionality reduction algorithms, such as Locality Sensitive Hashing (LSH) algorithm [15], to reduce data dimension and to address data similarities to store data efficiently.
  2- Semantic manager for patterns learning and semantics reasoning mimicking the neocortex functionality in the human brain
     - The reasoning operation depends on using: a) Associative Rule Learning (ARL) [21] algorithm and Fuzzy Membership Functions (FMF) [22] for recognizing data patterns via attributes discovery and classification, respectively, and b) statistical and probabilistic algorithms providing semantics reasoning models (Latent Dirichlet

Allocation (LDA) [23] and Hidden Markov Models (HMM) [24]) and behavior estimation models to extract network-semantics and to perform concept similarity matching operations.

3- Auto-associative storage memory comprising large extensible data tables as described in [25] for maintaining data/semantics. Storage memories of NMemAgents in lower NetMem levels possess short storage duration and they deal with highly dynamic data/semantics with lower levels of abstraction. On the other hand, storage memories of NMemAgents in upper levels maintain for long-term semantics with higher level of abstraction. Extracted semantics are kept as correlated groups of classified concept classes according to three network concerns, namely, Application, Communication and Resource (ACR) concerns as defined in [5]. Concept classification forms a DNCO that depends on the FBS engineering design framework [20]. The storage duration of maintained data/semantics in storage memories in the low level agents is shorter than the one of agents located in the upper levels. So, storage memories of agents in upper levels are longer-term memories compared with those of agents in low levels. Memories of agents in low levels mimic StM in the human brain while memories of agents in high levels mimic LtM in the brain. Shortest-term storage memories will be found in agents localized at the lowest NetMem level, while longest-term memories are the memories in agents located in the highest NetMem level.
4- Controller and Interface: a component that handles requests for data sent by various entities and sends control signals which state tasks of data discovery. Also, it is responsible for generating actions/alerts based on analytical reports received from components 1 & 2.

d. <u>Challenges for NetMem:</u>

Challenges in Data Management

- *Collection of voluminous data from heterogeneous sources* (<u>solution</u>: applying data virtualization techniques and data models to represent data uniformly and utilizing dimensionality reduction algorithms (LSH) to reduce data dimensionality
- *Learning patterns of heterogeneous dynamic raw network-data related to various Internet elements and building accessible ontology of network concepts* (<u>solution</u>: maintaining raw data and semantics with lower levels of abstraction in storage memories for short time data access and patterns learning and keeping extracted semantics with higher levels of abstraction in memories for long time semantics storage, access and matching processes.

- *Loss of data due to data dimensionality reduction that might affect accuracy of recollection process* (solution: change key length of hashing functions and/or number of hash functions adopted by the LSH algorithm)
- *Aging of maintained data that might affect quality and effectiveness of extracted semantics* (solution: change NetMem sampling rate for data profiles in general or related to specific network elements)

Challenges in Semantics Management

- *Discovery and selection of latent features* (solution: develop a hybrid semantics reasoning model integrating LDA and HMM for efficient semantics reasoning based on a) extracting high-level data features with long-range semantics dependencies and b) knowing associations amongst semantic topics related to various Internet elements)
- *Complexity of running NetMem algorithms (e.g., time complexity of some reasoning models and also at LSH with larger key lengths and hash tables' number, more storage space size would be required)* (solution: using hybrid reasoning models with LSH might enhance the time and space complexity of implemented algorithms)
- *NetMem overhead and efficiency which might affect networking operations (e.g., increasing network latency)* (solution: investigating optimum NetMem configuration to assign suitable reasoning models)

1.4 Network Scenarios with NetMem

We discuss in this subsection, the impact of utilizing NetMem over networking operations illustrated in the two scenarios presented in Section 1.2.

Learning QoS requirements of interesting and unfamiliar services

Figure 1.6 shows the effect of implementing the NetMem system (as multi-agent

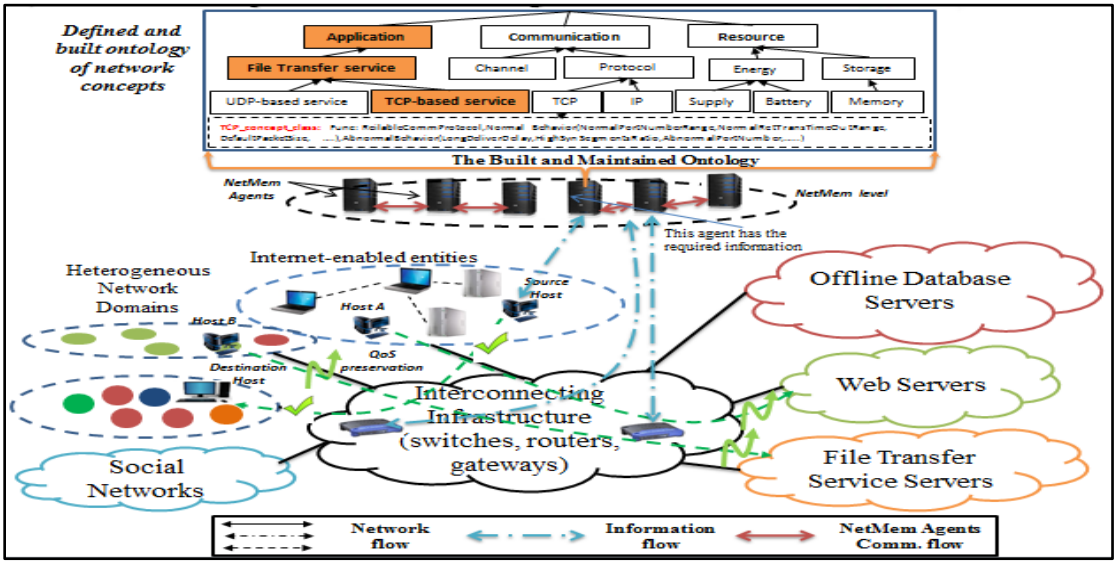


Figure 1.6 Learning QoS demands of interesting services via accessing semantics at NetMem

intelligent system) and the built and maintained ontology of network concepts. This ontology represents a clear definition for network concepts and concept classes related to various Internet elements (or network concerns) and at different levels of granularity showing the FBS aspects of each concept class. This ontology can be accessed by network entities at run-time and on-demand through a defined protocol used by those entities to communicate with NetMem agents. The scenario shows that interesting network entities (destination host, host A and host B, and routers) are able to get required information from NetMem's ontology at run-time. This aids in enhancing and learning QoS demands and concerns of running services.

Learning normal/abnormal behavior of services' flows

In case of having NetMem-based networking operations as depicted in Figure 1.7, intermediate routers are able to learn patterns and related semantics of running non-legitimate services. The routers communicate with NetMem agents requesting them to learn patterns of running services and clarifying the behavior of some interesting Internet elements (e.g., TCP and UDP protocols). NetMem might send the results of implemented reasoning processes which indicate the running services with anomalous flows are considered as classes of TCO-SYN flood and UDP-flood attacks. Once routers know that those traffic flows represent classes of some known attacks, they will suppress such flows to maintain QoS requirements of running legitimate services.

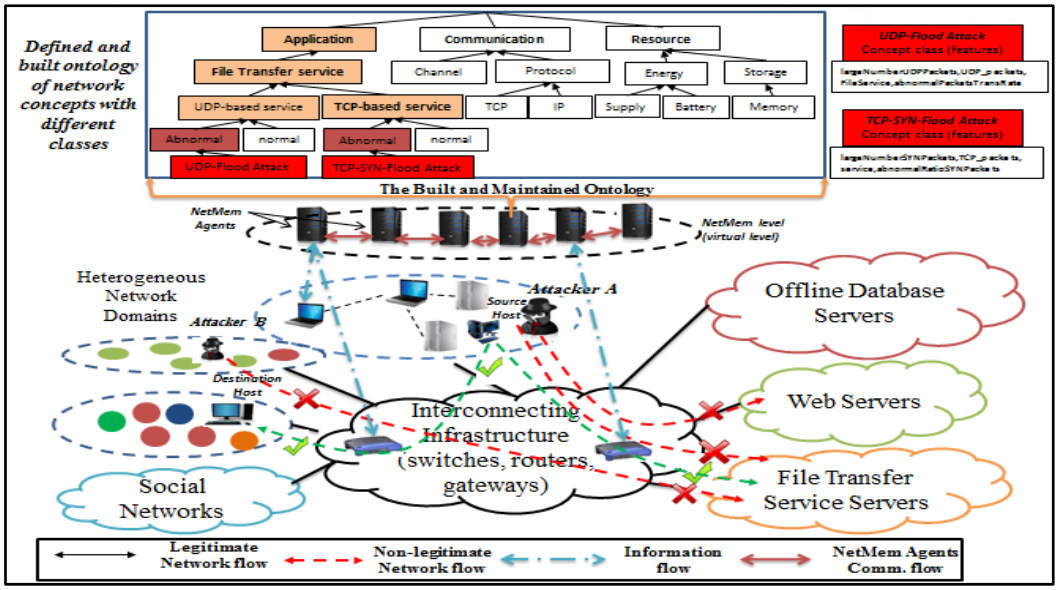


Figure 1.7 learning attacks in the network and mitigating their impact

1.5 Evaluation

We evaluated NetMem system efficiency and effectiveness. In order to evaluate the efficiency and demonstrate the scalability of NetMem as a multi-agent system comprising groups of communicating NMemAgents, we developed an analytical model. Furthermore, we formulated an optimization problem and a heuristic to help in reaching

near-optimum configurations using different semantics reasoning algorithms to improve NetMem efficiency. The effectiveness of NetMem was evaluated through simulation using real offline data sets and also via implementing a small practical test-bed over real Internet traffic data with and without using data dimensionality reduction techniques. We implemented NetMem as an application written in Java.

1.6 Contributions

Our main contribution in this dissertation is providing a methodology and a system for resolving the Internet Semantics Gap. We construct NetMem, a biologically-inspired distributed network memory management system for network-semantics. NetMem provides novel capabilities for network-semantics storage/retrieval/matching for supporting enhancement of many services, such as decision making and anomaly detection.

The results obtained by an analytical study showed the efficiency of NetMem as a multi-agent intelligent system to work with diverse reasoning models with full- and reduced-dimensional data. The hybrid semantics reasoner with the LDA-HMM technique succeeded in achieving the highest NetMem system efficiency compared with reasoning with HMM or LDA with/without using LSH. Additionally, there was a trade-off between NetMem system efficiency and enhancing the space complexity of the implemented semantic reasoning models using LSH.

Simulation results showed the effectiveness of NetMem to learn patterns of large-scale data with full- or reduced-dimensionality and reason about semantics that are used in a) enhancing QoS of running services; b) detection of anomalies and attacks; and learning normal/abnormal behavior classes of some Internet elements (e.g., services and attacks). Results showed the capability of NetMem to support existing networking tools (e.g., intrusion detection systems such as snort) to execute their functions more effectively and efficiently. In addition, the LDA-HMM-based reasoning model achieved higher NetMem effectiveness and efficiency whether using full- or reduced data dimensionality.

Intellectual Merit: We provide and evaluate NetMem system design and storage and reasoning methodologies for learning/representing/maintaining patterns of high- and multi-dimensional data related to various Internet elements. Our main contributions are as follows:

- Biologically-inspired customizable application-agnostic distributed network memory management system with efficient processes for extraction of classified high-level features and reasoning about rich semantics in order to resolve the Internet semantics gap and target Internet intelligence
  - Distributed system for efficient learning of patterns of multi-dimensional data with full- or reduced dimensionality and semantics with lower levels of abstraction and reasoning about semantics with higher levels of abstraction through providing:

        - Multi-level system with a set of agents at each level for supporting scalable semantics learning and management operations.
        - Memory system structure comprising:
            - Short-time Memory (StM): maintains for short-time raw data and semantics with lower levels of abstraction enabling learning patterns and extracting semantics with higher levels of abstraction; and
            - Long-term Memory (LtM): maintains for long-time less dynamic semantics with higher levels of abstraction and offers accessible correlated concept classes related to various network concerns for matching and prediction processes.

- Systematic methodology using monolithic and hybrid intelligence techniques for managing efficiently data semantics and building runtime-accessible dynamic ontology of correlated concept classes related to various Internet elements and at different levels of abstraction and granularity used, for example, in:
    - Predicting future events and learning novel things;
    - Recognizing and detecting normal/abnormal and dynamic/emergent behavior of various Internet elements; and
    - Satisfying QoS requirements with better utilization of resources.
- Analytical model and study for:
    - Evaluating quantitatively the NetMem system scalability and response time in the case of retrieving data and processing time in the case of data abstraction.
    - Reaching near-optimum configurations for the NetMem system, clarifying implemented semantics reasoning models, to maximize system efficiency (high semantics throughput).

Broader Impact: NetMem will provide:

- Support for semantics-driven and semantics-aware operations to enable "smarter" networking as follows.
    - Internet elements and tools will better understand traffic and element behavior in real-time, on-demand, and at low cost (providing or strengthening self-awareness, adaptation, and evolution capabilities).
    - Resolving the Internet semantic gap between network activities and the decision making process for better understanding of network dynamics.
    - Supporting existing networking tools (e.g., intrusion detection systems) to execute their functions more efficiently and effectively (e.g., enhance tools' prediction accuracy).

- Support for autonomic knowledge (network-semantics) discovery and responses via accessible and shared network concept ontology enhancing or providing:
  - Anomaly and emerging behavior discovery;
  - Efficient decision making capability due to semantics matching processes at different levels of granularity;
  - Dynamic QoS provisioning;
  - Resource utilization (e.g., in case of pre-allocating resources for unfamiliar services); and
  - Predictive operation (e.g., early detection and prediction of attacks).
- Providing dynamic network concept ontology related to various Internet elements and at different levels of abstraction showing FBS aspects of maintained concepts
  - Advancing network science and engineering to include evolution, modeling and simulation studies.

1.7 Document Organization

The remainder of this dissertation is organized as follows. Chapter 2 contains background and related work. A description of the NetMem system architecture and operations, such as data virtualization and data dimensionality reduction, is given in Chapter 3. Semantics management in NetMem is discussed in Chapter 4. The NetMem formal model and an analytical model for evaluating the NetMem efficiency is presented in Chapter 5. The effectiveness of the NetMem is evaluated via simulation over low and high volume of Internet traffic data in Chapter 6. Finally, Chapter 7 contains our conclusions and outlines and directions for future work.

Chapter 2

2 BACKGROUND AND RELATED WORK

This chapter covers concepts and techniques underlying our work. A survey of related work on network-data management and network intelligence is also included.

2.1 Background

2.1.1 Terminology

In this subsection, we highlight some definitions related to NetMem design as shown in Figure 2.1:

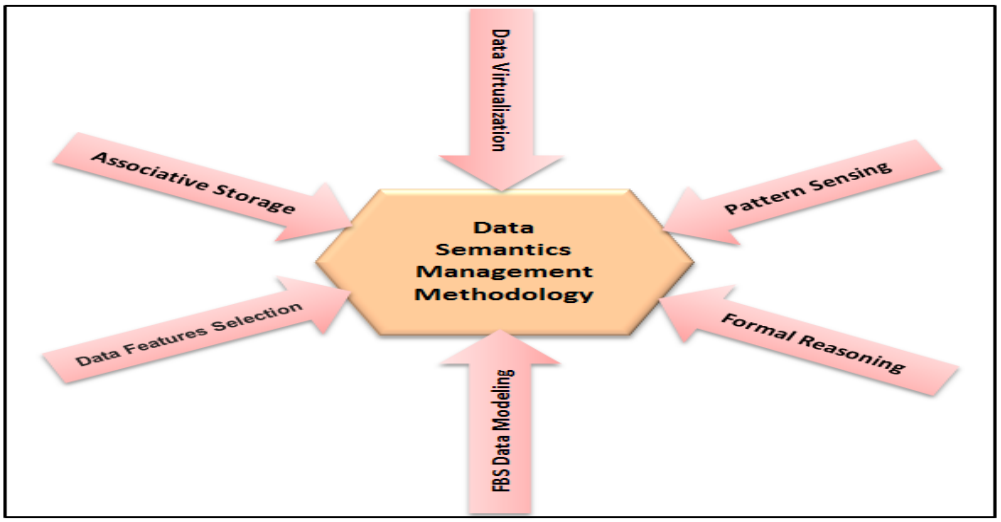


Figure 2.1 Data semantics management methodology

- *Big Network-Data:* data generated from different sources (e.g., Internet traffic, offline databases, management information bases (MIB)) with special characteristics, such as massive volume, information diversity (e.g., text, audio, video, etc.), high- and multi-dimensionality (e.g., large sets of attributes related to different Internet elements) and various dynamics concerns (e.g., time-sensitive data). With big network-data, there are challenges regarding content, structure and behavior.
- *Big Network:* a network that generates big network-data and can benefit from big data management in their operations. Examples of big networks include the current Internet, the emerging Internet of Things [9] and social networks.
- *Data Virtualization*: voluminous data are generated with different formats and representation modes from various sources in big networks. To have efficient data

collection, data virtualization (DV) techniques [26-29] should be used. This would enable data abstraction and federation from different sources employing unified data representation. We are inspired here by the sensory system in humans [16], which collects huge amount of data from five senses and sends it to the brain via nerve signals in a unified representation.

- *Data Feature Selection:* Big network-data are high- and multi-dimensional. This requires huge storage and computation capabilities to analyze patterns of these data. To have efficient big network-data processing, dimensionality reduction algorithms with the capability of directive data feature selection process will be used. Inspired by functionalities of the human memory, capturing raw data from various sources can lead to retrieving some distinguished features which are already maintained in the memory. For example, seeing an unfamiliar restaurant across a road aids us in easily remembering and retrieving some known special restaurant features like the existence of dining tables, chairs, restrooms, etc. (not the locations of restrooms or number and color of chairs and tables).
- *Function-Behavior-Structure (FBS) Data Modeling*: representing data uniformly, clarifying their functional, behavioral and structural aspects [19, 20] will facilitate data pattern learning. In the human memory, there are connectivity patterns established via synapses amongst neurons in different brain cortex areas. Those connectivity patterns refer to three different connectivity modes, namely, structural connectivity, functional connectively and effective connectivity between neurons. Structural connectivity gives information about the established links or synapses between neurons. Functional connectivity provides information about statistical dependencies and correlations between neurons. Effective connectivity refers to the information flow carried by electrochemical signals between neurons. The analysis of connectivity patterns using the previously discussed modes helps in learning the behavior and characteristics of neurons (e.g., specialized neurons) and related synapses and transferred electrochemical signals in different cortex areas.
- *Associative Storage:* to learn patterns, data will be maintained in warehouses, which will be extendible. That storage would be enabled with capabilities of identifying storage locations by their content or part of contents. This matches operations of short-term and long-term memories in human. Low or high levels of neurons fire in different cortex regions when they capture data, which indicate to different types of information. These groups of neurons are connected as sequences. They are referring to lots of detailed or abstracted data.
- *Pattern Sensing:* states the ability to discover big data patterns based on data attributes (or features) extraction and classification processes. In the human memory, sequences of neurons in certain cortex areas (e.g. visual and vocal areas)

with certain connection pattern lead to identifying characteristics of that pattern. For instance, hearing and seeing a cat lead to a certain neurons pattern in our brain based on learned concepts (e.g., expectation of listening to cat *meow* voice).

- *Formal Reasoning:* a well-founded artificial intelligence functionality based on integrated statistical reasoning models to perform semantic reasoning and matching. Those models will be used to extract semantics from learned data patterns and already known semantics. Constructed chains of neurons in different cortex locations (e.g., vocal and visual cortex areas) of the human brain result in formation of high level neurons that will be fired along the human life. Capturing data via different senses (e.g., sight and hearing) result in having sequences of correlated fired neurons in various cortex areas.

2.1.2 An Overview of Human Memory

The human memory as reported in [16] is capable of autonomously collecting huge amounts of data at different levels of details via our five senses (sight, hearing, smell, touch and taste). These data exhibit patterns that form connections between different areas in our brain cortex enabling travelling of nerve signals between groups of neurons or nerve cells. So, connection patterns among various locations of neurons form our memory. Captured data are represented in the human brain by a unified representation facility, which is called "nerve signals" carried by neurotransmitters through synapses between nerve cells. Data are coded in our brain's memory based on patterns of nerve signals, which vary in their strength and type.

The brain's cortex (or neocortex) comprises six layers and is responsible for forming the conceptual models in our memory. Lower areas in cortex face fast change signal patterns while patterns at upper areas experience slow changes. Formed signal patterns at upper cortex areas indicate invariant patterns with slower changes compared with those patterns at lower areas. These invariant patterns connect nerve cells in different cortex areas of various senses. The activity of those patterns like the strength of comprised nerve signals refers to specific information that might be updated and changed over the lifetime.

There are three main types of memory in our brain which are sensory memory, Short-term Memory (StM), and Long-term Memory (LtM) as described by the Atkinson–Shiffrin memory model [30] and shown in Figure 2.2. Memories in the human brain are built based on formed connectivity, through synapses, between nerve cells located in different cortex areas. This establishes signal patterns which relate data captured from different senses at lower areas of cortex to upper cortex areas. Hence, invariant patterns in the human memory combine fired nerve cells (i.e., cells have electrical signals) through sequences of nerve cells connected via synapses. The sensory memory receives data from the five senses and keeps data for a very short time (~ 1-3 seconds). StM and LtM in the human memory are established via areas in cortex. Both types of memory are formed based on nerve signals patterns that are created based on fired nerve cells due to traveling nerve signals. StM represents the lower areas of cortex which deal with data of

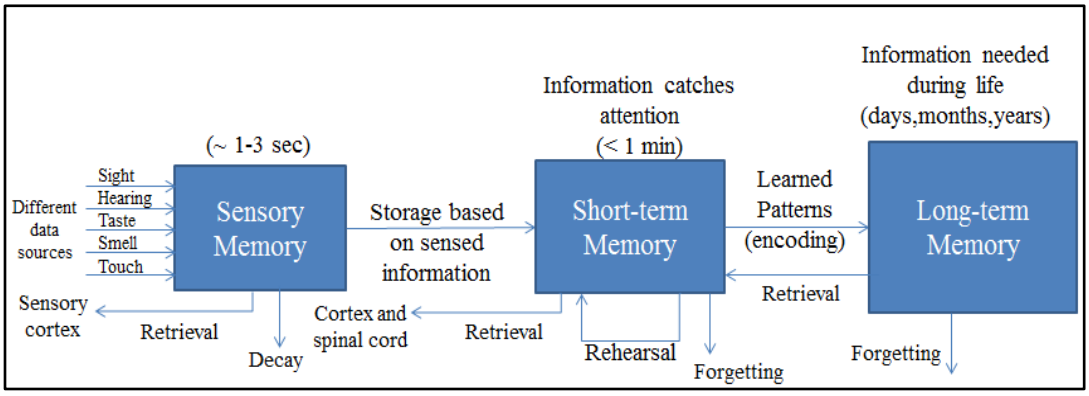


Figure 2.2 The Atkinson–Shiffrin human memory model

high rate of change. Data are maintained in StM for duration below one minute. LtM represents the upper areas of cortex which relates to data with low rate of change. Data might be retained in LtM for days, months, or years. StM and LtM depend on nerve cells and synapses connections among them. The human brain contains billions of nerve cells and they can be used based on signals received due to captured data from the various senses. There are dedicated cortex areas for each sense, however, some areas of a given sense can be utilized by other senses if not used and in case of high activity of captured data through human senses. LtM in the human brain comprises overlapped areas of different senses. These areas refer to maintained and invariant signal patterns which carry information gathered from various senses' areas at lower areas of cortex. So, when there are retrieval processes (e.g., remembrance of things or predicting behavior of novel things), the whole connected patterns are investigated including comprised nerve cells. This enables extraction of information related, for example, to vision or hearing. The invariant signals patterns formed at LtM due to input data patterns at StM are changed throughout life. Occurrence of known things with different behavior or learning novel things result in changes in invariant patterns established in the memory. Hence, human experience affects continuously formed patterns and their activity in the memory and consequently, the information they carry and refer to.

In our brain, data patterns are formed when: a) data are captured through the five senses; and b) data are retrieved from our memory (i.e., the thinking process). Patterns' rate of change refers to the activity of nerve cells and the rate of emitted nerve signals due to the amount, details, and novelty of received data. Patterns with low rate of change in the upper areas of the cortex represent semantics which are stored in nerve cells. This forms the human memory with long-term storage. Moreover, semantics in the human memory are accumulated as sequences related to nerve cells in various cortex areas. Semantics are updated continuously and can be associatively accessed and retrieved. The human memory possesses scalable storage through a tremendous number of nerve cells and the ability to relate nerve cells in different cortex areas (i.e., correlation of data from various senses). The data/semantics feedforward and feedback connectivity is established

in the human memory by synapses among neurons and to the motor system, which control motion of muscles. Based on maintained semantics, humans can predict future events, learn things and recognize new ones by matching their estimated semantics with those which are already registered.

2.2 NetMem as Network-Data/Network-Semantics Management System

2.2.1 Network-Data Management

2.2.1.1 Data Dimensionality Reduction Algorithms

There are various dimensionality reduction algorithms that can be utilized to reduce data dimensionality, such as Locality Sensitive Hashing (LSH) [15], linear discriminant analysis [31] and Principal Component Analysis (PCA) [32]. Table 2.1 shows a comparison between the previously mentioned algorithms.

Table 2.1 Comparison between different data dimensionality reduction algorithms

Algorithm	Description	Operation Technique	Application	Advantage	Limitation
LSH [15]	LSH is an algorithm for performing probabilistic dimensionality reduction of high-dimensional data. LSH can find similarities amongst huge amounts of data and support efficient data storage.	Bit sampling for hamming distance - Identifying a number of hash functions and the length of hash functions. - Calculating hash values for d-dimensional data. - Computing hamming distance according to defined set of hash function. - Finding similarities based on calculated and kept hash values in hash tables.	- Nearest neighbor search. - Hierarchical clustering. - Dimensionality reduction of high- and multi-dimensional network-data.	- Support high-dimensional data by hashing and detecting approximate neighboring points. - Support multi-dimensional data via the capability of applying directed data attribute selection processes through designing multi-hash functions for certain set of attributes. - LSH similarity search process leads to efficient data storage based on determining similarities	- High computation overhead in case of having large number of hash functions. - At large number of hash functions, large memory is required to maintain hash tables and related hash values.

Linear discriminant analysis [31]	It is algorithm which is used to reduce data dimensionality taking into consideration data class discriminatory information. It models differences between data classes and maximizes data class separability in order to find linear combinations of features that can best explain data.	- Based on certain data classes, fraction of data attributes or features related to each class are calculated. - Calculating the mean of each class and keeping variance of all data classes roughly constant (assuming: i) that the mean is the discriminant factor, and ii) Gaussian distribution of data).	- Statistics - Pattern recognition and machine learning. - Dimensionality reduction.	- Work efficiently with large number of data classes - Simple computation methodology.	- Assuming certain data distribution (Gaussian distribution). - Data overfitting - Challenges in case of working with small number of specific data classes and focusing on data features of interest.
PCA [32]	It is a statistical algorithm to transform using orthogonal transformation a set of correlated data variables or features into linearly uncorrelated variables (principal	- Using orthogonal linear transformation to have a new coordination for data variables. - Representing data	- Learning data patterns. - Dimensionality reduction. - Exploratory data analysis	- Support high-dimensional data. - Finding similarities and differences in data variables.	- Assuming input data have approximate normal distribution and data are real and continuous and

	components). The new orthogonal basis with axes oriented in the directions of the maximum variance of an input data set.	as vector of certain length and make transformation to represent specific principal component of data.			data can be reduced by linear transformation. - assuming that operation depending on maximizing variances will maximize information

2.2.1.2 Network-Data Management Systems

Some related work has provided management systems in order to enhance Internet operations by facilitating self-* properties and allowing protocol interaction through sharing information in a distributed knowledge plane (KP). The work in [33] proposed a distributed KP to the current Internet based on employing artificial intelligence and cognitive techniques to realize capabilities of self-knowledgeable, self-configuration, self-diagnosing, self-analyzing and self-managing. There was no management facility based on the proposed KP and no evaluation was provided. In [34], the authors proposed a centralized KP inside each communicating entity in mobile wireless ad-hoc networks. That KP helps entities in storing information related to protocols of the networking stack and other entities. The proposed knowledge system facilitates protocol interactions. Also, it requires periodic messages to update information registered in communicating nodes; and this causes additional communication overhead. Transmission power control aware protocols were designed to improve operation performance. In [35], Shieh *et al.* proposed a global federated KP for Internet where that plane provides trustworthy information regarding networking properties that will be used by running applications. The proposed KP was implemented via a group of servers operated by third parties to support security issues (e.g., confidentiality and privacy). There were defined access control policies which were used for information disclosure.

Researchers have investigated mechanisms for storing Internet and measurement data based on various attributes. For example, flexible meta-databases as in [11, 12] were proposed to enable accessing uniformly represented and correlated data from heterogeneous sources to support better data analysis and understanding of networking traffics. Cooperative Association for Internet Data Analysis (CAIDA) in [11] presented the Internet Measurement Data Catalogue (IMDC) as metadata repositories of measurement data to achieve smooth accessibility of that data for comparative analysis purposes. IMDC provided detailed information about stored data, such as its source and location and time of its occurrence. Authors in [12] proposed Scalable Internet Measurement Repository (SIMR) to track Internet measurements where large databases provide information about measurements, tools, users, experiments and datasets. These databases can be accessed easily for analyzing obtained measurements at various contexts and times. Similar to SIMR, the presented MOME in [36] is a web-based way for repositories of meta-databases independent of service types. Authors of MOME showed the capability to record information related to measurement tools and measurement data (as routing data and QoS attributes) where there is a public access capability, but processes of updating/retrieving entries are limited to registered users.

Some previous works have presented systems for managing and processing large sets of data, as in [37], to extract information that will be used to enhance decision making targeting specific goals. In [37], authors presented Apolo system to support sense-making for large network-data (e.g., data from social networks). Apolo depended on integrating

machine learning algorithm, called belief propagation, with user interaction and graph visualization technique.

Some models for semantics and behavior extraction are also found in the literature. Authors in [38] proposed unified Latent Dirichlet Allocation (LDA) model for categorization of web pages' semantics. They build that model through designing a separate LDA model for each category with specific urn of topic. Semantics inference process for unknown documents depends on results obtained each LDA model and using Gibbs sapling. In [39], Jiten *et al*. proposed a Hidden Markov Model (HMM) to support multi-dimensional data in image contexts and extraction of low level features. The proposed HMM model depended on Baum-Welch algorithm and maximum likelihood training to adjust HMM parameters and to find most likelihood semantic topics or features. Authors in [40] adopted HMM models and k-means clustering method for extracting human behavior. It constructs HMM from time series data using the Viterbi algorithm. It builds probabilistic density for frequency of occurrence "discrete" and successive time "continuous" using sequential discounting Laplace estimation and expectation and maximization [41]. Another system uses Support Vector Machine (SVM) for data analysis, classification, and pattern recognition plus Fuzzy rules [42] for discovering human behavior. It performs abstraction of data using SVM-based classification into sequence of actions. It constructs Fuzzy rules for each behavior, defined by a sequence of actions. Others use Neural Network (NN) and statistical approaches [43] to discover application behavior. It forms a stochastic model to represent application behavior variations using Markov chains. It builds a time series to represent application execution state changes then apply to a NN for predicting future behavior. The authors in [44] discussed two types of statistical models that enable prediction of future events and extracting information from large sets of complex data.

Researchers have also investigated mechanisms for data dimensionality reduction and semantics extraction in different fields, such as image processing and pattern recognition. Some works have used NN for extracting semantics [45] and reducing data dimensionality [46]. In [45], authors adopted NN to extract semantics or interesting events based on learning temporal video features of syntactic data segments. Multi-layer NN was used in [46] to form short-length codes for high dimensional image and document data based on weights assigned by network for extracted binary features.

Table 2.2 shows a comparison between NetMem and some related work for network-data management, such as CAIDA [11], Apolo [37] and other work presented at [12, 36].

Table 2.2 Comparison between NetMem and network-data management systems

Aspect / System	Goal	Design Approach	Impact	Limitation
Knowledge Plane for Internet [33]	• Attach self-* capabilities to current Internet by provide a distributed knowledge plane of high level models to have intelligent core.	• The design of knowledge plane has not been provided • Artificial intelligent and cognitive systems are recommended to be used	• Realizing self-* properties (e.g., self-configuration and self-diagnosing) to Internet-enabled entities enabling better QoS of running services.	• There are no abilities to extract and manage network-semantics based on maintained raw data and classified those semantics according to various network concerns. • No facilities for building dynamic behavior models related to various Internet elements
MANKOP [34]	• Facilitate networking protocols' interaction to meet dynamicity of mobile ad hoc networks as the high probability of topology change.	• Build a centralized knowledge inside each entity to store information about networking protocols	• Providing cross-layer protocol interaction in dynamic networking environments enabling enhanced operation performance at each communicating entity.	
NetQuery [35]	• Allow networking applications to access information related to network characteristics and properties	• Implement a knowledge plane by multiple servers operated by third parties to support confidentiality and	• Enhancing security for maintained and accessible network information.	

		privacy for registered information		
CAIDA [11]	• provide the ability to base predictions of Internet traffic, performance, and growth on real data rather than obsolete assumptions	• Internet data capturing tools and data models to gather and represent data uniformly at repositories and to derive parameters that will be used in analysis and generation of measurement reports.	understanding of the evolving commercial Internet infrastructure richer access of data to facilitate development of tools for navigation, analysis, and correlated visualization of massive network-data sets path specific performance and routing data that are critical to advancing both research and operational efforts offer suggestions to ISPs and routing vendors with respect to what instrumentation	

			within the router would facilitate diagnosing and fixing problems in real-time	
Apolo [37]	• Support sense-making for large network-data and enhance decision making.	• Integrating a machine learning algorithm, called belief propagation, with user interaction and graph visualization technique	• managing and processing large sets of data collected from social networks' data to extract information that will be used to enhance decision making targeting specific goals	
SIMR/ MOME [12, 36]	• Storing and sharing amongst researchers Internet measurement data.	• (No practical implementation for SIMR) and there are meta data models for MOME • Architecture: meta repositories for keeping data related to measurements, users, etc.	• Facilitate the sharing of data within the research community • For MOME (storing the collected meta-data about the measurement and monitoring tools) with GUI interface	
NetMem	• Provide capabilities for	• Distributed	• Runtime accessible	• Data semantics

	learning patterns of large-scale multi-dimensional network-data and recognizing network-semantics related to various Internet elements. • Share semantics among different networking environments and entities at run-time and on-demand in order to, for example, enhance networking operations and detection of anomalies	memory management system of data semantics mimicking the functionalities of human memory and comprising hierarchy of levels • Maintained data and semantics in agents are abstracted based on application, communication and resource network concerns and FBS aspects • Each NMemAgent integrates data virtualization and representation models, cloud-like data storage techniques, algorithms for data attributes discovery and classification, features extraction and classification	dynamic ontology of network concepts at different levels of abstraction and at different levels of granularity enabling: - Understanding of emergent/dynamic behavior of Internet elements. - Understanding behavior and tasks of unfamiliar elements. - Efficient decision making. - Predictive operation, (e.g., detecting anomalous behavior and attacks).	reasoning, discovery and retrieval from NetMem might result in time overhead which might affects network latency • Implemented semantics reasoning algorithms and constructed behavior extraction models might affect the accuracy and the level of abstraction of discovered concepts

		and reasoning and behavior estimation models.		

In all of the above-mentioned works, there are no methods for resilient and adaptable operation based on autonomic and explicit network-semantics management. We propose NetMem to address this gap by providing for network entities at real-time and on demand dynamic ontology of correlated network concepts related to various Internet elements.

2.2.2 Network-Semantics Management

2.2.2.1 Semantics Reasoning Schemes

In this subsection, we compare various semantics reasoning schemes that can be used in semantics management systems. Table 2.3 shows a comparison between different schemes for implementing models for semantics reasoning, such as HMM [24], LDA [23], neural networks [47], Latent Semantics Analysis (LSA) [48], Simulated Annealing (SA) [49] and Support Vector Machine (SVM) [50]. The comparison highlights the advantages and the limitations of each technique. More details will be discussed for HMM and LDA in Chapter 4.

Table 2.3 Comparison between different schemes for semantics reasoning

Scheme	Description	Operation Technique	Application	Advantage	Limitation
HMM [24]	- HMM is a statistical Markov model for categorical sequence labeling (i.e., pattern recognition by statistical inference) based on using supervised/unsupervised learning algorithms (e.g., Baum-Welch algorithm [51]). - Sequence labeling can be treated as a set of independent classific	Based on continuous input sequence with different Gaussian distributions then making distribution mixture for obtaining most likelihood output sequence.	- Labeling documents/motions with certain tags or topics for information retrieval - Gesture imitation in robots	- HMM depends on a mathematical model with parameters) that can be adjusted for supporting different semantic topics in many contexts - HMM's statistical foundations are computationally efficient and well-suited to handle new data - HMM can support multi-dimensional data	- The floating-point underflow problem. - discovering high-level features with long-range semantics dependencies - lot of HMM parameters to be calculated at large input states - require large sets of data to be trained - Low performance at operation with reduced-

	ation tasks. - HMM is a structured architecture that is able to predict sequences of semantic topics based on hidden sequences of input data attributes or features.			- HMM can be adjusted as prototype to extract spedific semanti topics	dimensional data
LDA [23]	- LDA is a generative probabilistic dynamic model that can be used for extracting hidden topics in group of observed unlabeled input data profiles and known words or attributes. - LDA has the capability to discover high-level features in data profiles where it has a well-defined inference methodology to associate a data	Based on supervised learning process and training data with prior probabilities that associate topics with words and documents and using sampling tech. (e.g., Gibbs)	- Modeling a mixture of topics with documents - Topic models for image and text recognition	- LDA models are extendible to support more associations amongst semantic topics and data attributes - Can be integrated with functionalities of other semantic reasoning models (e.g., HMMs) - discovering high-level latent features	- randomization process for assigning parameters' values of attribute-topic - Probability for the overfitting problem at large number of training data sets - Big bag of attributes lead to topic misclassification process based on random topic sampling process

	profile with group of attributes with several semantic topics. - LDA is capable of extracting data semantics based on prior probabilities for data profile-semantic topic and attribute-semantic topic associations.				
LDA-HMM (Hybrid) [52]	Probabilistic and Categorical sequence labeling supervised/ unsupervised algorithms for allocating latent topic and estimating semantics based on hidden sequence of input latent words	Based on sampling process for hidden topic bases on multinomial probability distribution and using Gaussian distributions with maximum likelihood estimation for outputting semantics	- Topic modeling for many applications such as pattern and image recognition, and information retrieval	- Combine advantages of both LDA and HMM schemes - Number of input features or states to train the HMM is lower than operation with HMM alone - Mitigate the effect of topic misclassification and overfitting by LDA where final semantics output depends on group of syntactic states formed by extracted features by LDA	- The time complexity worsens to some extent compared to comprised monolithic schemes - The floating-point underflow problem. - Require large sets of data to be trained

Simple statistical-analysis-based models	Models for learning statistics of input words and based on some defined rules, they can extract semantics	Collecting statistics about words, documents and using defined rules (e.g., Fuzzy rules) and thresholds that outputs depend on meeting those rules	- Rule-based Reasoning (extraction of information related to specific semantic topics)	- Can be designed to be directed to specific types of semantic-topics or information → achieving good performance - Low computation complexity	- specific to certain semantic topics - Inefficient design lead to incorrect extracted information - No capability to extract high-level latent features
Neural network [47]	Classification algorithms (supervised algorithms predicting categorical labels) - learning is by training and used to model complex relationships between inputs and outputs, to find patterns in data, or to capture the statistical structure in an unknown joint probability distribution between observed variables	-Based on a hidden layer that relates group of input with other group of output via defined weights. -connection weight based on relation between inputs and outputs and affect results (i.e. outputs) -Depending on composition of functions where components of each neural network layer are independent of each other	- System identification and control - Decision making - Pattern recognition - Data mining	- flexibility: it can be applied to many applications - robustness: it can work properly and mitigate the failure of some Internet elements - It can handle noisy data and analyze complex data patterns	- Design complexity: 1) complicated group of neural units connected based on weighted inputs to outputs and passing thresholds 2) long processing time in case of large neural network - more trainings is needed to operate efficiently - Its operation might face overfitting

Latent Semantic Analysis (LSA) [48]	Probabilistic models which relates documents to topics based on weighted relationship specified by probabilities (mapping documents' words into concepts)	-Building models for mixtures of topics and documents through training examples	- Modeling a mixture of topics and concepts with documents and their words	- Reduces dimensionality of documents for better representation, finding synonyms and minimizing computational complexity.	- Assuming Gaussian distribution for words in documents. - No directed capabilities for selecting efficient dimensions of reduced documents (based on heuristics)
Simulated Annealing (SA) [49]	Its name is inspired from annealing in metallurgy. It is a Probabilistic metaheuristic algorithm for searching for global optimum solutions of objective functions and with large search space.	Its operation methodology depends on a slow decreasing in probability for accepting worse solutions and moving from state to another state depending on a defined acceptance probability function, generated new solutions by mutation and a generated random number for making decision through number of iterations or until getting no more better	- optimizing operations of online information searching through massive number of Internet web pages and related information	- Capability to operate with arbitrary problems (e.g., combinatorial problems) and systems. - Easy to be coded and implemented for complex problems.	- Slow according to its sequential nature. - One candidate solution per iteration and capability for building quick overall view of search space. - Overstated for problems with few local minima.

		solutions.			
Support Vector Machine (SVM) [50]	Machine learning models depending on supervised learning algorithms. SVM is used for analyzing data and learning patterns	Non-probabilistic binary linear and non-linear classifier using numerical quadratic programming It depends on a set of training examples via support vectors (to build an assignment model) and data input where it will classifies each input to one category	- Data classification - Anomaly detection - Regression analysis - Feature and online searching - Text and hypertext categorization	- Easy for training - No local minima - Versatile through using different kernel functions for modeling complex problems and decision making	- Poor performances at long feature dimensions compared with number of samples or support vectors - High complexity (long training time) with extensive memory requirements at large scale tasks - Selection of appropriate kernel functions to suit problems.

2.2.2.2 Network-Semantics Management Systems

In this subsection, we compare between our solution for knowledge and semantics management and other related work, such as [53-55]

Table 2.4 Comparison between NetMem and knowledge/semantics management systems

Aspect / System	Description	Architecture	Management methodology
Legal Knowledge Management System (LKMS) [53]	• Collaborative web-based platform (using semantic web technologies) for knowledge management, supporting law firms in managing a document base and the processes around it	• integrating a semantic layer (SemantiK), founded over a centralized ontology repository, into an existing knowledge management environment • SemantiK provides a middleware for a high-level access using RDF language and semantics search and integration	• building knowledge (semantic web) repositories and have ontology handler to analyze knowledge queries • applying Fuzzy logic for measure of closeness to search for required information which is maintained at different repositories
Semantic Manufacturing Knowledge Management System [54]	• Semantics-based computer-aided manufacturing knowledge management for realizing knowledge-intensive manufacturing • management system combines knowledge engineering and semantic web	• Semantics-based knowledge solidification framework, using semantic agents, running over the Internet for managing and developing manufacturing knowledge • Practical example: knowledge query system development with the support of semantic search engine and RDF technology	• Manufacturing knowledge standardization (computer-aided modeling process) • Manufacturing knowledge instantiation (computer-aided process for making knowledge instances) • Manufacturing knowledge query (a computer-aided process of acquiring knowledge)
Knowledge Management	• ontology-based semantic web	• designing a conceptual	• sharing knowledge via

via Ontology-based Semantic Web Services [55]	services framework management system to improve integration and performance of services and application across the Internet	model (framework) for knowledge management with the adoption of a set of semantic web services ontologies using OWL-S language • four main communicating layers: user interaction, interface, monitor, and ontology	various distributed sources (distributed repositories) • management among sources: 1) discover candidate web services, 2) select the most appropriate, 3) mediate any mismatches, and 4) invoke the selected web services
NetMem	• human memory's functionalities-based management system of data semantics related to various network elements and on different levels of abstraction • Semantics are maintained at NetMem, which is shared by different networking environments and entities at run-time and on-demand in order to, for example, enhance networking operations and detection of anomalies	• logical shared distributed multi-agent intelligent system • a hierarchy of intelligent agents located in virtual levels based on the degree level of data abstraction • each agent is embedded with algorithms for data virtualization, dimensionality reduction, patterns learning, semantics reasoning	• learning and classifying data features based on analyzing attributes at raw data using associative rule learning and Fuzzy logic • Recognizing high level latent features using latent dirichlet allocation algorithms • utilizing hidden Markov models to reason about data semantics based on having group of latent features

2.2.2.3 Semantics Ontology Formation

In this subsection, we highlight the difference between our NetMem solution with already existing solutions for semantics ontology formation as presented in Table 2.5. Research in [56] and [57] investigated building ontology of semantics support semantic web services and knowledge about network attacks mitigation, respectively. In [56], the author presented a methodology for building ontology through extracting data semantics from web pages and forming ontology of web concepts that enable enhanced QoS of running web services and applications. The ontology is built by experts and updated by web users. The ontology is formed via utilizing distributed intelligent agents which can access using ontology languages different small web ontologies located on separate network entities or resources. Agents can communicate together to exchange required information and users can access terms defined at ontologies via browsing tools, languages and rules.

Undercoffer *et al.* [57] proposed a reasoning system (Backward-chaining and Forward-chaining reasoners) to reason about knowledge over data sets and forming a knowledge base. Authors targeted defining ontology of terms concerning network attacks (e.g., denial of service attacks) to aid in detecting attacks and improving the performance (e.g., minimize false alarm rates) of intrusion detection systems. Their designed ontology presents data abstraction level includes classes and sub-classes refer to systems, attacks, processes and including attributes and properties. Updates occur in the formed ontology based on changes in the studied data sets and the outcome of the implemented reasoners.

Table 2.5 Comparison between different approached for semantics ontology formation

Aspect / Ontology	Broader impact	Building methodology	Data abstraction level	Access, Communication and Update
Semantic web ontology [56]	- Extracting data semantics from web pages and forming ontology of web concepts that enable enhanced QoS of running web services and applications	- Using distributed intelligent agents which can access via languages (DAML+OIL) different small web ontologies located on separate network entities or resources	- Providing properties with various options (values) based on each required service and available ontologies (resources) which can	- Network entities can access terms defined at ontologies via browsing tools, languages and rules - Agents can

		- Ontologies are built by experts and developed by web users - Applying mapping techniques and pointers to link between ontologies - Establishing a hierarchy of linked classes and sub-classes of services' properties	offer those options	communicate together to exchange required services' properties - There are links between ontologies that updating entries of services in an ontology can be reached by other ontologies
Network security attacks ontology [57]	- Defining ontology of terms concerning network attacks (e.g., denial of service attacks) to aid in detecting attacks and improving the performance (e.g., minimize false alarm rates) of intrusion detection systems	- Using a reasoning system (Backward-chaining and Forward-chaining reasoners) to reason about knowledge over data sets and forming a knowledge base - the ontology is built using the specification language DAML+OIL	- Defining linked classes and sub-classes based on network hosts - classes and sub-classes refer to systems, attacks, processes and including attributes and properties	- Defining a protocol for knowledge query/respond based on DAML+OIL language - Updates occur in the ontology based on changes in the studied data sets and the outcome of the implemented reasoners
Network concepts ontology by NetMem	- Extracting data semantics related to various Internet elements (services, protocols, attacks, etc.) and building dynamic ontology of network concepts on different level of granularity showing FBS aspects for better understanding network	- Human memory-based AI functionalities (patterns learning, semantics reasoning) - Using distributed intelligent agents to extract/maintain semantics on different levels of abstraction relied on captured raw network-data with low levels of abstraction.	NetMem system is designed as a hierarchy of levels which comprise agents. Each level describes a specific level of abstraction for maintained data/semantics or	- Network entities can access network concepts at run-time and on-demand via a communication protocol - Agents can communicate via a semantic

	dynamics, QoS enhancements of networking operations, better resource utilization	- Forming a hierarchy of associated concept classes related to various Internet elements and representing those classes using XML DTD language	concepts in agents at that level where maintained data/semantics are classified according to three network concerns (application, communication, resource) showing FBS aspects of each concept	query/response protocol to exchange/update required concept classes - Agents in different NetMem levels might leave an update flag concerning interesting concepts

2.3 NetMem as Intelligent System

Some works targeted intelligence-based solutions to enhance operation performance in different fields (e.g., networks, speech and image recognition). The following subsections discuss some solutions using monolithic and hybrid intelligence techniques for achieving intelligence in different areas, such as speech recognition, language modeling and networking.

2.3.1 Intelligence-based solutions for speech and pattern recognition and language modeling

In the literature, some works have targeted intelligence-based solutions to enhance operation performance in different fields (e.g., speech and image recognition). In [58], authors integrated HMM and Neural Networks (NN) to form a hybrid speech recognition system. They employed NN to extract discriminative speech features by processing multiple instances of the same feature vector. Extracted features are then directed to HMM to model the acoustic behavior of speech. HMM helps overcome NN's limitations in extracting valuable information from highly-dynamic traffic data. Another system uses support vector machine (SVM) for data analysis, classification and pattern recognition plus Fuzzy rules [42] for discovering human behavior. The system performs abstraction of data using SVM-based classification into sequence of actions, and constructs Fuzzy rules for each behavior, defined by a sequence of actions. In [52], the authors provided a system for language learning based on using LDA and HMM. They adopted the ability of LDA and HMM to simultaneously learn and find syntactic classes and semantic topics despite having no knowledge of syntax or semantics beyond statistical dependency.

2.3.2 Intelligence-based solutions for Internet Intelligence

Internet (or network) intelligence (referred to here as InetIntel) is defined in the literature as the capability of Internet elements to understand network-semantics to be able to make effective decisions and use resources efficiently [14]. InetIntel has to support Internet elements with the capability for learning normal and dynamic/emergent behavior of various elements and in turn building dynamic behavior models of those elements. Consequently, this will enable elements to be conscious of surrounding contexts enabling them to enhance their performance, utilization of resources and QoS satisfaction.

InetIntel might use deep packet inspection mechanisms to extract data packets and their content to: i) learn information about different types of IP traffic and Internet elements, such as services, applications and protocols, and ii) expect occurrence of events. InetIntel provides facilities to understand network traffic by identifying correlation among users, services, and protocols; and it might be able to represent acquired knowledge in a unified model. InetIntel is considered as a middleware where it forms an information layer with metadata from IP traffic. These data are fed to applications to enrich their information about network-based activity. Furthermore,

InetIntel relates data from different traffics to enhance situational awareness and better cyber security and IP services. In business intelligence and marketing, InetIntel can enhance network monitoring and analysis by capturing at real-time information about different services and applications that users are interested in them. Also, through the IP layered stack, InetIntel can classify running protocols and applications among layers to generate metadata that can be used in correlating activity between layers and enhancing protocols interaction and operation. To achieve its targets, InetIntel maintains some tools, as provided by [6], such as performance analytics tools (e.g., flow data analysis) and security analytics tools (e.g., intrusion detection systems), inline acceleration tools (e.g., traffic compression appliance) and inline security tools (e.g., data loss prevention appliance). Here are some of the services that can be offered by InetIntel:

- Optimization for QoS of running services and applications and enhancing protocols operation based on end-to-end (e2e) and non e2e principles.
- Unified representation "metadata" for different types of traffic to be used by applications.
- Real-time traffic analysis and situational awareness.
- Behavior analysis based on statistics, for various Internet elements, such as users, services, applications and protocols.
- Accumulated and evolvable knowledge services within time for better decision making processes in different situations as anomaly discovery.

InetIntel can be achieved via employing intelligence techniques to design intelligence systems. Those systems will reason about semantics from Internet traffic data and provide for networking entities accessible valuable information on different levels of granularity. InetIntel should be achieved in a way that will not affect negatively the Internet robustness and scalability. Intelligence systems for InetIntel can work as a stand-alone system or to be integrated with the current Internet infrastructure (e.g., Internet gateways and DNS servers). Some works [7, 59-62] have been investigated for InetIntel. For example, the work in [61] aimed at enhancing QoS of Internet-based web services. The authors provided intelligent Internet agent based on using hybrid simulated annealing technique. That agent enabled efficient methodology for interesting information searching through the massive amount of ever-increasing Internet information.

InetIntel systems can employ monolithic and/or hybrid intelligence techniques. Each implemented monolithic technique for InetIntel has its mechanisms for learning data patterns, extracting features and reasoning about data semantics. The environment of Internet has tremendous and ever-growing scale. It is noisy and dynamic with dissimilar communicating networks and heterogeneous entities, running services and resources. Accordingly, generated and transmitted Internet data have special characteristics, such as massive volume with high- and multi-dimensionality. Those characteristics might affect negatively performance of monolithic intelligence techniques. Hybrid intelligence techniques (HIT) can mitigate that challenge [63, 64]. HIT will integrate more than one

monolithic intelligence technique. HIT can mitigate limitations of monolithic techniques that it can combine their significant capabilities to extract valuable information with good level of accuracy and timeliness. For example, in [65], comparisons among various InetIntel techniques-based intrusion detection systems showed the outperformance of HIT in achieving higher detection accuracy.

In the networking literature, monolithic- and hybrid intelligence-based solutions have been investigated to provide intelligence for networks and Internet to realize self-* properties, address intrusion detection and achieve improved network performance and operation. We classify InetIntel systems according to their addressed solutions, as shown in Figure 2.3. We classify NetMem as a hardware-software based solution for InetIntel since NetMem is an application that depends on data captured from various sources (e.g., from Internet traffic through sensory systems using remote physical or logical sensing end units [66]).

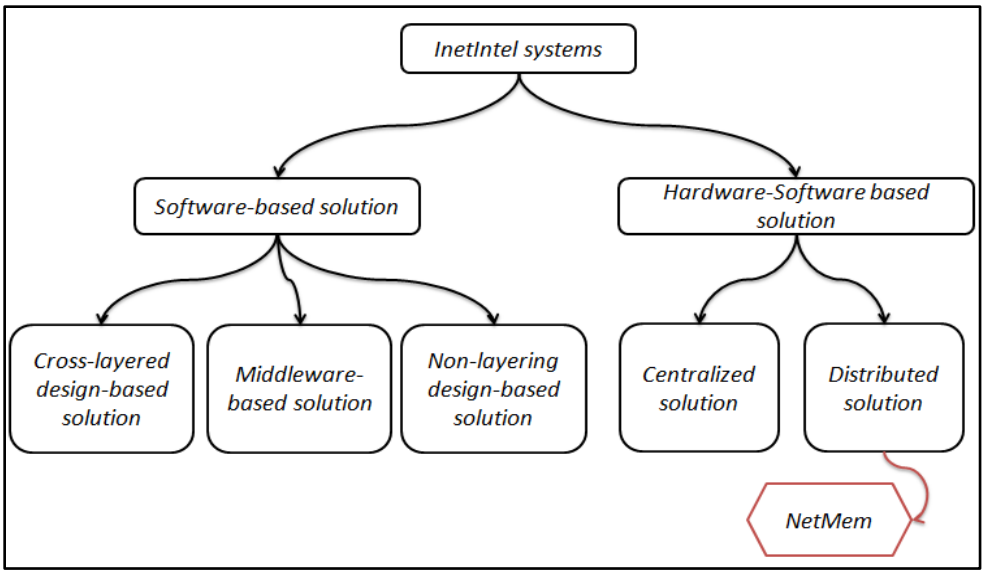


Figure 2.3 Taxonomy of Internet intelligence solutions

1- *Software-based solution:* that solution depends on injecting for networking architectures new processes and techniques via codes in already implemented systems to enhance/add functionalities and features to those systems. Monolithic intelligence techniques and HIT are an example of techniques that can be attached to networking entities to embed intelligence to networks for enhancing their services. Proposed works at [61] and [67] are examples of that class. We further classify software-based solution for InetIntel into three sub-classes:

   a- *Cross-layered design-based solution:* It is a solution for InetIntel based on layered networking architectures and their comprised components. The idea is to provide functionalities and algorithms in each layer that allow interaction between all layers and feedback to provide more self-* properties. Cognitive networks [68] are an example of that class.

b- *Middleware-based solution:* that solution depends on layered networking architectures where an information layer is augmented with functionalities and algorithms that enhance interaction between all layers and feedback to provide/strengthen self-* properties. The presented work at [59] is an example of that class.

c- *Non-layering design-based solution:* that solution relies on non-layered networking architectures where functionalities are added to enhance networking operations among entities and to strengthen their self-* properties (e.g., environment awareness).

2- *Hardware-Software based solution:* that solution introduces an embedded system of interconnected and cooperated components, which are programed to perform assigned tasks. Proposed systems based on that solution might adopt intelligence techniques to achieve InetIntel. We further classify hardware-software based solution for InetIntel into two sub-classes:

a- *Centralized solution:* that solution provides embedded systems which are installed and implemented in a central entity that can perform functions at different domains and can be shared by other parties. For instance, hardware components (e.g., chips) with implemented algorithms are added to a specific system level in a networking entity to allow it to behave in an intelligent way. The discussed works at [7],[65] and [69] are an example of that class.

b- *Distributed solution:* that solution introduces embedded systems are installed and implemented in a distributed way (e.g., over set of communicating entities or agents) that can perform functions at different domains and can be shared by other parties. The proposed works at [60] and [70] besides NetMem are example of that class.

2.3.2.1 Solutions for realizing self- properties*

Authors in [71] proposed a reasoning system, called Pellet, that supports self-learning and self-awareness capabilities for network entities in Internet concerning web semantics. Pellet provided web ontology language-Description Logics (OWL-DL) reasoner for extracting web semantics Pellet was designed so that different tableaux algorithms [72] can be plugged in. Pellet is able to learn and maintain concept classes related to web semantics in order to enhance web pages and related contents discovery and retrieval. Also, Pellet is capable of extensive support for reasoning with individuals (including nominal support and conjunctive query), user-defined data-types (e.g., XML Schema based data-types), and debugging support for ontologies. Additionally, it supports the standard array of derivative queries and various reasoner management services both via its application programing interface (API).

2.3.2.2 Solutions for enhanced security

Hybrid intelligent techniques were proposed to embed intelligence to operation networks for enhancing their services and security. Kumar *et al.* in [65] discussed artificial intelligence-based techniques that can help in enhancing intrusion detection in Internet. A classification for those techniques was provided. Machine leaning and genetic algorithm based techniques are examples of artificial intelligence-based techniques. In [69], Idris *et al.* provided a software-based middleware solution via a HIT for intrusion detection system (IDS). The implemented HIT comprises Fuzzy logic and a data mining mechanism with the usage of neural networks (NN). The IDS system depended on extracting features or attributes from large sets of real network-data and applying those features over simple *if-then* Fuzzy rules. The proposed system combines: a) misuse detection (e.g., detect attacks based on learning patterns and matching with already known attack patterns); and b) anomaly detection (e.g., learning unfamiliar attacks or threats by applying statistical analysis methods over data and compare results with historical knowledge).

In [73], the authors provided an intelligent IDS using software-based solution which integrates Fuzzy rough sets and decision trees based on using C4.5 algorithms [74]. The proposed system targeted mitigating new emerged attacks in the ever-increasing of current Internet usage. The proposed system depended on group of designed static Fuzzy rough rules. Khan *et al.* [75] provided another software-based solution for enhancing security in sensor networks via adopting HMM for discovering abnormal temporal events in those networks. The assumption was that unusual events are rare and not enough data are found for training. Different detection models were implemented where abnormal activities were detected if their likelihood were below defined thresholds.

According to survey of IDS works in [64, 65, 69, 76], there were no facilities to form and maintain accessible ontology of network concepts based on learned data patterns and known data semantics. NetMem has a main difference which is the capability of building dynamic ontology of concepts clarifying behavioral models of different Internet elements. Those models can clarify the different classes of attacks and/or the abnormal flows of various services. This can help networking built-in tools (e.g., IDS) to learn behavior of attacks' classes and anomalous flows. Consequently, this will lead to enhancements in IDS's performance (e.g., increase prediction accuracy and minimize false negatives).

2.3.2.3 Solutions for enhanced network performance and operations

Some work, e.g., [7], offered hardware-software based solutions that provide chips to be installed on networking entities. Those chips are responsible for analyzing traffic relied on header-based packet classification besides payload pattern searching. This work provides an autonomous capability for entities to learn patterns in data traffic without the need for external memories. Some capabilities are found as monitoring network performance and capacity utilization. In addition, there is a possibility, relied on specific traffic profiles, for behavior analysis and availability of applications using packet header

fields and payload patterns. From real-time analysis results, there is a facility to take reactions as blocking or rate-limiting some traffic profiles and sending alerts. The work can measure some service quality metrics as delay, jitter and throughput. Furthermore, it provides API for accessing live and historical data. Similar to some ideas in [7], the author in [70] showed network intelligence in mobile environments by analyzing traffic through inspecting at real-time packets' header and payload and using signature patterns, stored in a database, to be used in matching streams of packets. This methodology can generate some statistics that lead to study behavior of applications, protocols and users. For instance, usage patterns for a certain application can be built which show its usage times over the day and its link capacity utilization and number of users who are interested in it. Based on learned patterns and triggered events, some charging policies and rules might be installed and activated. Applied policies (e.g., routing and security policies) will suit running applications behavior and user demands and type.

In [59], the author proposed an interaction system to achieve security network intelligence that would target security end-to-end communication. This system provides interconnecting and functionality coupled architectures to enable interaction and communication among different functional levels. Also, the system shows a hierarchical structure for a network comprises different types of nodes that can cooperate to enhance QoS. Other trials as in [60] targeted a mobility management architecture based on core network intelligence not just end-to-end intelligence. Nodes in core, not just at end, with different capabilities have to cooperate to enhance QoS for highly dynamic infrastructure networks. This work contradicts the classic principle initiated by the Internet Engineering task force document RFC1958 which states that the goal is connectivity, the tool is Internet protocol and the intelligence is end-to-end rather hidden in the network.

Authors in [76] explored a NN model which integrates supervised NN (e.g., self-organizing map) and unsupervised NN (e.g., multi-layer perceptron) to efficiently discover attacks. Yao in [77] proposed a software-based solution for a HIT to provide evolved NN with enhanced performance. The author integrated NN with evolutionary algorithms (e.g., genetic algorithms [78]) to update different NN domains, such as weights in distributed connections, architectures, learning rules and input features. That evolved NN model would enable operation of that artificial intelligence based technique to detect events in dynamic environments (e.g., mobile ad hoc networks). These works did not possess the ability for learning data patterns and extracting valuable information or semantics from highly-dynamic Internet traffic data with high-dimensionality.

Other software-based solutions of middleware type using HIT were offered in [61, 67] for enhancing network performance regarding different network concerns like information routing and discovery. Authors in [61] provided a hybrid Simulated Annealing (SA) algorithm for optimizing operations of online information searching through massive number of Internet web pages and related information. The proposed algorithm can perform its intended tasks without human intervention. That algorithm

depended on automatic textual analysis of Internet documents instead of using keywords or hypertext linkages. However, hybrid SA algorithm-based search engine has no capabilities to extract information based on learning features or semantics in Internet documents. In [67], a cross-layered-design based solution via a HIT comprising Hopfield Neural Networks (HNN) and SA was proposed for developing an efficient routing algorithm for communication networks and Internet. SA was used to provide HNN's optimal parameters and to reduce HNN's number of iterations to converge efficiently. The proposed method aimed at minimizing the computation cost of HNN. The proposed HIT depended on SA technique with its generation process of random solutions to optimize HNN's operation. There was no capability for extracting network-semantics (e.g., behavior of routing protocols in wired/wireless links among nodes) and using those semantics to enhance the operation of routing algorithms.

2.3.2.4 How NetMem Differs?

Unlike the aforementioned application-specific solutions for achieving InetIntel (e.g., [67, 70, 71]), NetMem presents a customizable application-agnostic way for autonomously managing network-semantics and targeting InetIntel. Also, NetMem provides a storage memory structure which comprises short-term memory and long-term memory to facilitate pattern learning of data with lower levels of abstraction and semantics reasoning/retrieval for matching and prediction processes. Furthermore, NetMem offers a distributed solution for scalable reasoning operation via a set of reasoning agents at multiple levels. This suits characteristics of network-data and provides various levels of data abstraction. Furthermore, NetMem's distributed architecture and agents at different levels enables specialization of agents to extract specified concept classes related to specific networks concerns (e.g., some agents may be dedicated to extract concept classes related to behavior of communication protocols). Previously proposed solutions for enhancing IDS [64, 65, 69, 76] did not provide capability for forming accessible dynamic ontology of concepts based on learning patterns of high- and multi-dimensional network-data with massive volumes.

Derived concept classes by NetMem can be accessed and learned by Internet elements at runtime and on-demand enabling, for example, learning new things (e.g., QoS requirements of unfamiliar services) and predicting future events (e.g., attacks). Accordingly, this will aid in improving networking entities' performance, security (e.g., detecting in advance anomalies) and attaching an efficient situational awareness capability to entities to optimize/stabilize QoS of end-to-end (e2e) and non-e2e services. Table 2.6 highlights the differences between NetMem and other solutions for intelligence using monolithic and hybrid intelligence techniques.

Table 2.6 Comparison of Solutions for Netowrk Intelligence

Solution	*Added Capabilities*	*DataSets*	*Intelligence*	*Computational Complexity*	*Operation Scalability*	*Functionality*	
						Application Specific	*Data Availability*
NetMem (LDA-HMM-based intelligence system	- Memory system design for intelligence: 1) The ability for feeding raw data and semantics with lower levels of abstraction into memory for enabling a place for short time storage of highly dynamic data showing various concerns and learning data patterns 2) Forming ontology of concept classes with higher levels of abstraction	There are network-data sets with the following characteristics: -Massive volume -Dynamicity -High- and Multi-dimensionality - Yield patterns	- Structuring a memory system with functionalities similar to the human memory (StM and LtM) - Learning patterns of raw data and semantics with lower levels of abstraction - Then, extracting data semantics and storing them as concept classes	For LDA: O(MKN) where M is the profiles number, K number of semantic topics, N is number of attributes per profiles For HMM: $O(K^2T)$ where K is the number of semantic topics and T is the length of input state sequence The overall complexity is O(polynomial)	- Distributed agents over multi levels enabling communication with large number of raw data sources - Different number of agents per each level for having the capability of handling high volume of varying data and also volume of less dynamic data semantics or concept classes	**Application-Agnostic Customizable** - Wide scope extracted semantics related to various network concerns - Capability to have customizable concept classes or application-specific ontology related to specific network concerns (e.g.,	- Highly dynamic raw data are maintained for short time enabling semantics reasoning models for learning patterns and extracting semantics - Little dynamic or varying data (semantics) are available for long time

	(related to various network concerns) at long-term storage memory showing FBS aspects of each class and providing capabilities for matching and prediction - The hybridization of LDA and HMM to combine the advantages of both algorithms in extracting variable-length classified features sequence related to various network concern and reasoning about associated semantic topics based on that features		- Enabling access stored concept classes at runtime and on-demand in matching and prediction processes		- Multi levels for representing different levels of data abstraction and various storage durations for abstracted data (i.e., StM and LtM) showing the hierarchy of associated classes with different levels of granularity	ontology of communication protocols)	- Runtime accessible ontology of various associated concept classes related to different network concerns

		sequence - Ability for building dynamic continually-updated ontology with a facility of customizable formation according to interesting real-world applications or network concerns (e.g., certain Internet applications)						
Monolithic Intelligence Technique-based intelligence systems	**Simulated Annealing (SA)-based solution (e.g.,** [61])	- optimizing operations of online information searching through massive number of Internet web pages and related information - Output most relevant web	Network-data with: -Massive volume -Dynamicity -High- and Multi-dimensionality - Yield patterns	Classifying analyzed web pages based on their interesting attributes and maintaining them in a long-term database	Polynomial time (NP-hard and NP-complete)	- The proposed solution seems to be centralized where the solution depends on one agent which interacts with one homepages'	**Application Specific non-Customizable** - Specific for applications of relevant web pages searching (semantics	- Known homepages and related features are kept in a long-term database

		pages				database (it might face long response time at huge amount of homepages' information at the database)	web)	
	SVM-based solution (e.g., [42]**)**	- Data analysis, classification, and pattern recognition plus fuzzy rules for discovering human behavior - Abstraction of data using SVM-based classification into sequence of actions, and constructs Fuzzy rules for each behavior, defined by actions' sequence	Data characteristics: - Big volume - Dynamicity - Yield patterns - low-dimensionality	Learning behavior patterns and storing those patterns in a repository for future usage like in matching processes	The standard SVM has computational complexity of $O(m^3)$ where m is the training set size	The proposed solution seems to be centralized where sensory data are collected to a single chip as appeared in the presented prototype	**Application-specific customizable** Directed to detect human behavior and there is a capability to focus on specific actions and related behaviors based on defined fuzzy rules and trained SVM and implemente	Extracted and defined data patterns (actions and behavior) are stored in repositories

							d sensors	
	Hidden Markov Model (HMM)-based solution (e.g., [75]**)**	- Implementing HMM models with capabilities of training on and knowing usual human activity events (combining events in one class) and detecting unusual events based on rare and not enough data if related data features' likelihood were below defined thresholds - Alerts based on each input discriminative reading	Data characteristics: - Big volume - Dynamicity - Yield patterns - low-dimensionality	Differentiating between usual and unusual human activity events based on trained HMM models over the usual events	$O(K^2T)$ where K is the number of semantic topics and T is the length of input state sequence	- The proposed solution seems to be centralized where readings are collected to a single location (agent or engine)	**(Application Specific Customizable)** Specific for detecting unusual events (e.g., readings) in data acquisition networking environments (e.g., sensor networks))	- Just alerts based on decisions which clarify usual and unusual events. - No built behavioral models for usual and unusual events which can classify the different types
Hybrid Intelligence Technique-based intelligence systems	**Fuzzy logic and Neural Network (NN)-based solution (e.g.,**	- Extracting features or attributes from large sets of real network-data - Reducing the dimensions of	Network-data with: -Massive volume -Dynamicity -High- and Multi-	- Generating alerts based on if-then Fuzzy rules and extracted data	For neural networks: O(polynomial) like $O(n^k)$ where n equals number of inputs and k is constant	- The proposed solution seems to be centralized - This will possess	**(Application Specific Customizable)** - Specific for application	- Previously made decisions and cases can be stored for

	[69])	extracted features through self-organizing maps with unsupervised learning - Generation of alerts based on applying extracted features over simple if-then fuzzy rules - Decisions might be stored for future usage at similar cases	dimensionality - Yield patterns	attributes. - Performing matching processes with already maintained decisions		inferior operation performance with high raw data volume	s of anomaly based and host based intrusion detections	future usage and matching processes
	Pellet [71]	- OWL-DL reasoner with extensive support for reasoning with individuals (including nominal support and conjunctive query), user-defined data-types (e.g., XML Schema based data-	Web data with: -Massive volume -Dynamicity -High- and Multi-dimensionality - Yield patterns	learning concept classes related to web semantics to enhance web pages and related contents discovery and retrieval	The worst case complexity of tableau algorithms is o(exponential)	The proposed solution seems to be centralized	**Application-specific customizable** It targets semantics web ontology however there is a capability for customizing output	Extracted semantics are maintained as ontology with hierarchy of concept classes in accessible web pages

		types), and debugging support for ontologies - support the standard array of derivative queries and various reasoner management services both via its own API - The reasoner is designed so that different tableau algorithms can be plugged in					ontologies with concept classes based on implemented tableau algorithms and defined requirements (integration with rules)	
	LDA-HMM based solution (e.g., [52]**)**	- Ability to simultaneously learn and find syntactic classes and semantic topics despite having no knowledge of syntax or semantics beyond statistical	- Huge vocabulary volume - High dimensionality - Complexity (e.g., different languages and	Language modeling and document classification through a learning capability for words with short-range and long-range	O(polynomial)	The proposed solution seems to be centralized where data are collected from different corpora to a central reasoner	**Application-agnostic customizable** The solution might be used for many applications such as language	There is no indication about how to get obtained semantic classes and syntactic classes

		dependency - Generative model that adopts short-range syntactic dependencies or long-range semantic dependencies	representation models) - Variety in information (different topics, figures, etc.)	syntactic and semantics dependencies			modeling (part-of-speech tagging) and document classification	
	Hardware-software based systems (e.g., [70])	- network intelligence in mobile environments by analyzing traffic - Traffic analysis process was performed via network monitoring tools which inspect at real-time packet header and payload using signature patterns - patterns stored in a database and used in matching	Network-data with: -Massive volume -Dynamicity -High- and Multi-dimensionality - Yield patterns	Storing learned patterns in a database and used those patterns in future matching processes and taking decisions	Not available	- The proposed solution seems to be centralized where learned signature patterns are stored in a centralized databases - This will possess inferior operation performance with high raw data volume related to various	**(Application Specific Customizable)** - The presented work shows application-specific operation where signature patterns are built for mobile networking environments - Signature patterns related to	- Learned data patterns (including for example typical packet length, uplink/downlink packet rates) are stored in a database for future usage and matching processes

		streams of packets				network concerns and accordingly a lot of patterns can be learned and stored	various applications or network concerns might be adopted (application-agnostic operation)	

Chapter 3

3 NETMEM: BIOLOGICALLY-INSPIRED NETWORK MEMORY MANAGEMENT SYSTEM

In this chapter, we present our distributed network memory management system architecture showing its hierarchical structure as a distributed multi-agent intelligent system, functions and operations. We also present the architecture of NetMem agents, or NMemAgets, and discuss data management in NetMem.

3.1 NetMem Architecture

We propose a distributed multi-agent system architecture for NetMem, as shown in Figure 3.1. Such distributed realization [79] will achieve the following goals: a) runtime accessible dynamic network-concept ontology (DNCO) with different levels of abstraction; b) scalable operation for continual ontology growth; and c) minimizing network-data and concepts management overhead.

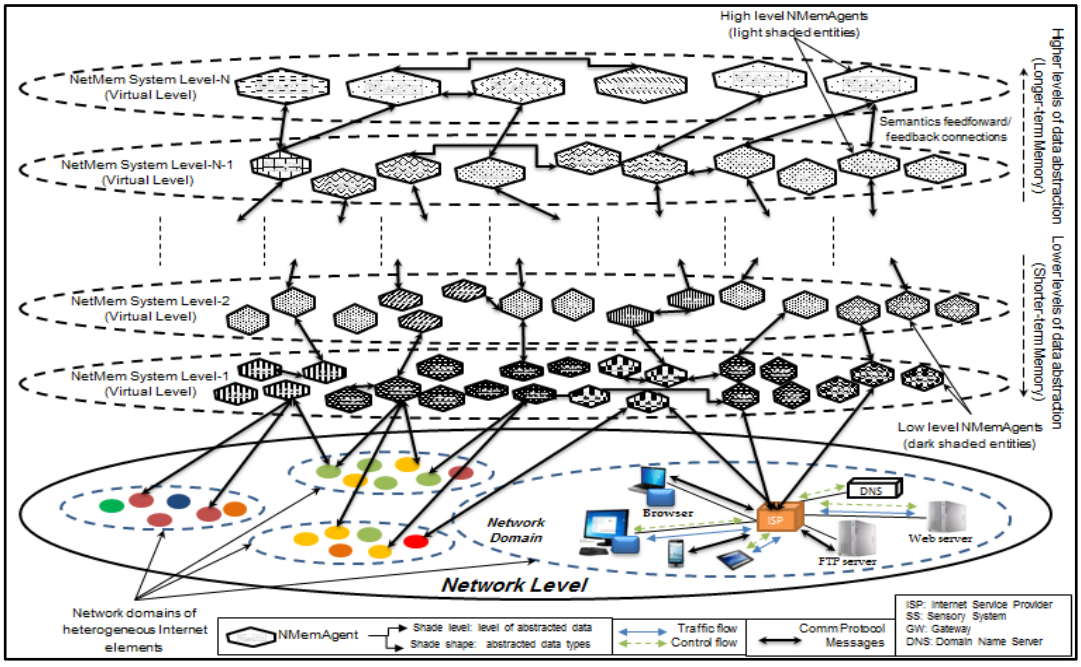


Figure 3.1 NetMem as a distributed multi-agent system

NetMem agents, called NMemAgents, are placed in a hierarchy of levels whereas each level contains a set of agents. Each NMemAgent at a specific level has its own

capabilities and storage memory for: i) maintaining network data/semantics gathered from network entities or agents at lower levels where these data/semantics are at lower levels of abstraction, ii) learning data patterns and extracting semantics with higher levels of abstraction, iii) registering data/semantics at different levels of granularity for facilitating semantics matching and behavior learning processes, and iv) communicating with other agents at the lower/same/upper levels enabling semantics learning, retrieval and updating.

Each NetMem level represents an abstraction level for maintained data/semantics defined by the content of the storage memory in each NMemAgent. Each level has a higher level of data abstraction than the level below it. Additionally, the storage duration of concepts maintained in upper NMemAgents is longer than that for concepts in lower NMemAgents. So, the behavior of storage memory in agents at lower levels resembles StM, while it is closer to LtM in agents at upper levels. The hierarchy of concepts extracted by NMemAgents at different NetMem levels and with various abstraction levels enables building DNCO related to various network concerns. Internet elements (e.g., hosts) in various networking environments can access and request at runtime and on-demand extracted information in the formed DNCO via communication with NMemAgents at the lowest level.

For semantics management and DNCO building, NMemAgents adopt machine learning algorithms, feature extraction and classification algorithms and reasoning models to learn patterns and high-level features of data/semantics at low levels and reason about semantics with higher levels of abstraction. For DNCO updating, NMemAgents run a behavior estimation model to update the DNCO via investigating differences between newly extracted semantics and existing semantics. In addition, agents adopt the estimation model to search for and retrieve semantics requested by other agents at different NetMem levels.

3.2 NMemAgent Architecture

As shown in Figure 3.2, a NMemAgent comprises a set of composable and cooperating building blocks forming processes of i) data feed-forward for patterns learning and semantics reasoning, and ii) semantics feedback for matching and prediction tasks. NMemAgent main components are:

a) Memory: an auto-associative storage memory for maintaining data/semantics where it comprises relational extensible big data tables as described in [25]. Those tables are a form of cloud-like data storage mechanisms [80] which is called cloud table storage mechanism. There is a capability in [25] to store large datasets in big tables based on an open source implementation technique for big tables for massive scalability defined in hbase [81] and built on top of the Java framework hadoop [13]. Storage memories of NMemAgents at lower NetMem levels possess short storage duration and they deal with highly dynamic data/semantics with lower levels of

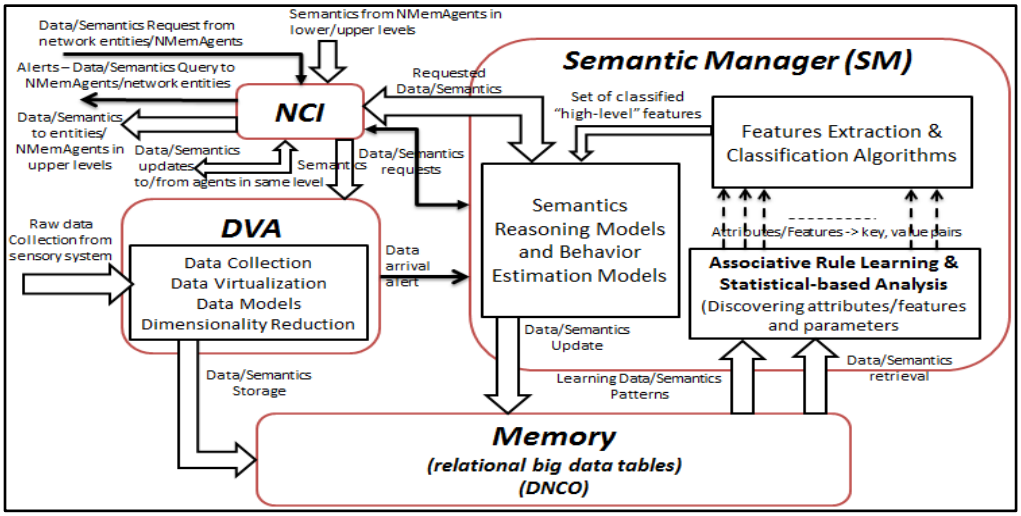


Figure 3.2 NMemAgent Architecture

abstraction. Memories of low level agents mimic lower cortex areas in the human brain and they maintain data and semantics with lower levels of abstraction. On the other hand, storage memories of NMemAgents at upper levels maintain for long-term semantics with higher level of abstraction. Memories of high level agents mimic higher cortex areas in the human brain where they deal with invariant or less varying semantics patterns. Data attributes and features are maintained according to three network concerns, namely, application, communication and resource (ACR) concern as defined in [5] and using a framework for concern separation as defined at [82]. Semantics with higher abstraction levels are kept as correlated groups of classified concept classes according to ACR concerns. Concept classification forms a dynamic network-concept ontology (DNCO) that depends on the FBS engineering design framework [19, 20] where functional, behavioral and structural aspects are defined for each concept. FBS aspects of each registered concept show: 1) the functional aspect: type and roles of semantics, 2) the behavioral aspect: shows for instance if extracted semantics refer to normal and abnormal behavior, and 3) the structural aspect: illustrates correlations among semantics of various Internet elements. The storage duration of maintained data/semantics in storage memories in low levels' agents is shorter than the one of agents located at upper levels. So, storage memories of agents at upper levels are longer-term memories compared with those of agents in low levels. Shortest-term storage memories will be found in agents located at the lowest NetMem level, while longest-term memories are the memories in agents located in the highest NetMem level.

b) Data Virtualization and Access (DVA): mimics sensory memory system in human [16]. DVA performs data collection, acquisition and modeling operations for registering data/semantics in the storage memory enabling patterns learning. On one hand, DVA in NMemAgents located in the lowest NetMem level implements a) DV techniques [29] for data homogenization via using data models to unify

representation (as profiles of attribute-value pairs) for raw data collected via a sensory system from various sources with different formats; and b) dimensionality reduction algorithms (e.g., LSH algorithm [15]) for high-dimensional raw data reduction, grouping, and similarity matching processes. On the other hand, the DVA role in NMemAgents in the upper NetMem levels is to collect semantics with well-known format from neighboring agents and represent semantics in the storage memory as associated concept classes related to various network concerns;

c) Semantic Manager (SM): mimics the neocortex functionality in the human brain [16]. SM executes data semantic reasoning operations. SM tasks depend on learning patterns of data/semantics kept in the storage memory. Based on learned patterns, SM will: i) reason about semantics with higher levels of abstraction, i) update contents of the storage memory based on learned semantics, and iii) send extracted semantics to agents in the upper level. Figure 3.3 shows pseudo code of a skeletal pattern learning and semantic reasoning processes. SM learns groups of attributes in data profiles or features of concepts maintained in the storage memory through utilizing associative rule learning (ARL) algorithms [21, 83, 84] and simple statistical-analysis-based models using fuzzy membership functions (FMF). SM adopts features extraction algorithms and reasoning algorithms (e.g., LDA [23] or HMM [24]) for learning data semantics with high levels of abstraction. SM has behavior estimation models to be able to update kept semantics in the storage memory based on new discovered semantics. This enables NMemAgents to predict future events and learn behavior and requirements of new Internet elements. Data models utilized by both DVA and SM show FBS aspects of maintained data and semantics (or concepts) using the FBS engineering design framework. SM can begin

```
Data packets (data/semantics) = Sense(traffic,NCI_port); // data collection by NMemAgent DVA via sensory system and protocols through NCI
Profiles (p) = dataModeling(data/semantics); // unifying data representation as profiles of attribute-value or feature-value pairs
for all p_n ∈ all n profiles p
startDataDimensionalityReductionAlgortithm (p_n); //reduce dimensionality of data in case of agents in the lowest NetMem level
    if similarity(p_n) > threshold then; // check similarity based on various criteria as targeting large number of similar attributes
    storeToStorageMemory(p_n);
        end if
end for
sendSemanticManager(alert); // DVA alerts SM that the storage memory is updated with data/semantics
for all p_m ∈ m profiles p in stroage memory at reasoning time period t
    A_m = runAssociativeRuleLearning(p_m); // SM extracts attributes per profiles A_m = {a1, a2, ....., ad} where d is profile dim.
    assignProfileNumberMembership(p_m); // SM uses Fuzzy membership functions to classify number of similar profiles
        F_m = startFeaturesExtractionAlgortithm (A_m); // SM extracts sequence of T latent features F_m = {f1, f2, ....., fT} for every profile
        O_m,t = runSemanticsReasoningModel(F_m); // SM extracts sequence of most likelihood T observations O_m = {o1, o2, ....., oT} for every profile
        S_m,t = generateSemanticTopics(O_m,t); //SM defines semantic topics based on obtained observations
    S_m,t-1 = checkLastSemanticTopics(p_m); //SM detects previous semantic topics defined for profiles
    for each semantic topic or concept class i related to profile m
    if S_m(i),t == S_m(i),t-1 (true positive or negative) then
    C=defineConceptSequence(S_m(i), O_m(i), F_m(i)); // SM constructs a concept based on extracted features and observations
        c_m = runAssociativeRuleLearning(C); // SM extracts from the storage memory concerns per stored concept C= {c1, c2, ....., cT}
        matchingResult = similarityMatch(c_m(i), O_m(i), F_m(i))
        if matchingResult == false then
        sendUpperAgents(C); // SM sends extracted concepts to upper agents
else    sendUpperAgentsAndUpdateLocalStorageMemory(C); // SM sends upper agents and updates existing concepts in the local memory
        end if
        end for
end for
```

Figure 3.3 NMemAgent operations for patterns learning and semantics reasoning

its work according to alert signals sent by DVA which indicate to arrival of data/semantics to the storage memory; and

d) Controller and Interface (NCI): responsible for handling requests about data/semantics sent by Internet elements in various networking environments and neighboring NMemAgents at same, lower and upper NetMem levels. NCI sends requests to SM to seek information in the storage memory. In case of SM inability, NCI will send information requests to neighboring agents in the same or the upper NetMem level. SM can ask NCI to search for missing data/semantics in order to enrich in the storage memory built ontology of associated concept classes with more classes and increase levels of abstraction. NCI is embedded with protocols that enable it to communicate with other agents and entities. Also, it adopts Simple Network Management Protocol (SNMP) [85] to discover data registered at Management Information Bases (MIB) [18] located at network entities. Based on results of analyzed patterns and extracted semantics by SM, NCI receive reports and it might generate actions as sending alerts to entities outside NetMem. Those alerts clarify, for example, that: 1) entities' data/semantics requests were processed successfully (or they were denied); and 2) semantics with behavior similar to known network attacks have been identified.

3.3 NMemAgent Communication Protocol

Many languages (e.g., KQML/KIF [86] and FIPA-ACL [87]) were produced to provide standardization for machine-readable representations to help in web semantics markup. Annotation of semantics in web contents provides richer information that help in better understanding web sites and contents and enhancing QoS and benefits gained from the various web services. NMemAgents obey to Foundation for Intelligent Physical Agents (FIPA) standards for heterogeneous and interacting agents and agent-based system FIPA standards:

- Agent management and Agent Communication Language (ACL) specifications.
- Performatives that can be executed by agents.
- Query/response message formats.

a- Ontology Query Response Protocol (OQRP)

We provide Ontology Query Response Protocol (OQRP) for communication among NMemAgents and between agents and network entities in network domains. OQRP is an application-layer protocol for NMemAgents to collect: 1) raw data about various Internet elements; 2) data/semantics from other NMemAgents at different NetMem levels. OQRP messages have the capability to be addressed based on multi-criteria where this descriptive addressing facilitates messages processing by network entities or NMemAgents. Whether NMemAgents or network entities at the network level can specify in their data/semantics OQRP request messages certain information-based criteria based on which responding entities/agents will feedback available information. (for

instance, agents might send group of network entities requesting them to send it back maintained information about some aspects of a network entity with certain IP address like running services, used communication protocols, etc.). In case of collecting raw data from network entities and similar to SNMP, OQRP can retrieve information from MIB [18] found in network devices and it will work on top of UDP protocol.

OQRP is different from SNMP [85] that: 1) it does not manipulate data maintained at MIB; and 2) it enables NMemAgents to communicate together to get information about specific network concepts in already built ontology. MIB maintains structured raw data which maintains information about network devices' or entities' objects where each object has unique identifier called object identifier (OID). MIB's information will be represented as encoded Abstract Syntax Notation one (ASN.1) [88] using XML [89] encoding rules to achieve better human readability. MIB provides a standardized data structure mechanism for representing data about heterogeneous Internet elements manufactured by different vendors. This enables NMemAgents to get, discover and analysis data about multiple elements, such as communication protocols. For example, NMemAgents might want to learn behavior of normal TCP protocol in specific networking contexts. Through the standard OID of TCP protocol at MIB which is "1.3.6.1.2.1.6", NMemAgent can get and learn TCP data in various network entities. Low-level TCP OIDs from "1.3.6.1.2.1.6.1" to "1.3.6.1.2.1.6.15" give information about several TCP statistics and attributes like minimum/maximum retransmission timeout.

Ontology Query Response Protocol (OQRP) based on FIPA-ACL message formats

The OQRP query/response message format defines the following fields, as shown in Table 3.1:

Table 3.1 The message format of NMemAgent communication (OQRP) protocol

Source Header	Destination Header	UDP Header	Msg Code	Msg ID	OID(s)/Criteria Profile/ Concept Class(es)	Content	Ontology	Language	Checksum	TTL

1. Source Header: this field represents the IP addresses of source (up to 16 bytes) and source agent identifier, if it exists, which includes number of the related NetMem level where that agent is implemented
2. Destination Header: this field represents the IP addresses of destination (up to 16 bytes) and destination agent identifier, if it exists, which includes the related agent's NetMem level.
3. UDP Header: which defines the source/destination port and this field is of length 4 bytes where each of the source/destination port has length of 2 bytes
4. Msg code: this field shows the message type/agent performative whose size is of length five bits and based on list of performatives by FIPA-ACL:
   - "00000" query message when NMemAgents require information about certain OID(s) from network entities;

- "00001"- "10100" (i.e., 1 to 20) messages with a certain performative of 20 defined ones such as:
  - ♦ "01101" query message when NMemAgents send other NMemAgents, in the same level, underlying level, and next top level, requesting ***updates*** about specific concept class(es);
  - ♦ "01000" response message when NMemAgents ***inform*** other agents due to query messages of certain IDs
  - ♦ "00100" call for proposal message when NMemAgents communicate with other NMemAgents ***requesting*** help for doing specific tasks regarding certain concept class(es) with/without specifying certain aspects (e.g., functional, behavioral or structural aspect) and constraints (e.g., maximum storage period);
- "10101"- "11110" message codes are reserved for future (e.g., defining more performatives or control functions)
- "11111" response message when network entities ***response*** to query messages with certain IDs about OID(s).

5. Msg ID: it is integer value field of length one byte which represents the ID of the message
6. OID(s)/Concept(s) Profile(s)/Concept Class(es): this field is of variable length and it will be:
   - OID(s): sequence of integer in case of type "00000" message (i.e., requesting information about certain objects).
   - Criteria Profile: that profile, in case of requesting information or negotiation messages, describes criteria on which data/semantics updates will be sent. Each criterion is described as a key/inputs/description triple. The key shows criteria identifier (e.g., IP or concept class name); the inputs shows aspects related to that criterion and it is a vector of text strings; and the description provides interpretation for the relation between a criterion and its inputs. The description might be Boolean, real numbers or text strings (i.e., requesting information (data/semantics) based on multi-criteria).
   - Concept Class(es): Text strings in case of type requesting information messages (i.e., requesting specific concept class(es)).
7. Content: it is of type string with variable length where it contains description, in case of query messages, about required OID(s) or concept class(es) clarifying concept's functional, behavioral and structural aspects. It also contains information in case of response messages about, for example, required semantics or about agreed/refused action based on sent proposals.
8. Ontology: it is of type string with variable length where it contains description about the used ontology determining the hierarchy of concept classes with respect to certain network concerns. This field can be found in query/response messages.

9. Language: this field is of type string with variable length and it gives information about language used to describe contents in the content field. For example, FIPA standards might use semantic language (fipa-sl) or Prolog.
10. Checksum: this field is for error checking data where the field is calculated using the source/destination headers plus the content of required OID(s) or concept class(es).
11. TTL: time to live field is integer of length one byte which is set by data/semantics NMemAgent source (i.e., TTL is set in response messages). This field is used to improve the performance of caching by discarding entries in update tables and semantics/data contents stored in memory that exceed the TTL value.

b- Concept-based Communication Protocol (CCP)

We design a concept-based communication protocol (CCP) for communication and data/semantics transfer between NMemAgents and network entities. CCP is an application-layer protocol and it will work on top of UDP protocol to minimize overhead and complexity (stateless protocol), and to enhance delay of data retrieval. CCP is used by network entities (e.g., hosts) for retrieving information from NetMem about interesting Internet elements (e.g., services, protocols, etc.) which relate to application, communication, and resource network concerns. Entities will send messages through CCP to NMemAgents requesting knowledge about specific elements. Entities can determine required aspects of those elements. For instance, an entity can request information about the function of UDP protocol. Once NMemAgent can meet that request via having required information, it will send a response. The CCP request/response message has the following fields, as depicted at Table 3.2:

Table 3.2 The message format of the concept-based communication protocol

IP header	UDP header	Msg type	Msg ID	Concept class(es)	F	B	S	Value	TTL

- IP header: this field is up to length of 32 bytes and it represents the IP addresses of source (up to 16 bytes) and destination (up to 16 bytes).
- UDP header: which defines the source/destination port and this field is of length 4 bytes where each of the source/destination port has length of 2 bytes.
- Msg type: this field shows the message type whose size is of length two bits and it equals
  - "00" alert message when NMemAgents send alerts to network entities concerning specific concept class(es). (e.g., attacks).

  - "01" update message when a network entity sends a message requesting continual updates regarding information of certain concept class(es) from NMemAgents.
  - "10" request message when a network entity sends a message requesting information of certain concept class(es) from NMemAgents.
  - "11" response message when a NMemAgent responses to request about information of certain concept class(es).
- Msg ID: it is integer value field of length one byte which represents the ID of the message.
- Concept class(es): this field is of variable length and of type "string", and it will describe required concept class(es).
- F, B, and S: three fields describe if functional (F), behavioral (B) and structural (S) aspects of required concept classes are needed specifically. Each field is of length one bit and if it is set in request messages, this means that requester is interesting in that field (i.e., concept class aspect). If a request message is sent without setting any of those fields, NMemAgents will respond based on available information about requested concept class(es). In case that a requester needs specific aspect of concept class(es) and NMemAgents cannot find that aspect, it will communicate with other NMemAgents using OQRP protocol searching for that aspect.
- Value: it is of type string with variable length where it contains description about required concept class(es) clarifying concept's functional, behavioral and structural aspects if needed specifically. This field is of length zero in case of the query message.
- TTL: time to live field is integer of length one byte which is used to improve the performance of caching by discarding old messages.

3.4 Server Hosting NMemAgent

In NetMem, NMemAgents can implemented applications or services over network servers that can communicate using a communication protocol as defined by FIPA-ACL [87] for inter-communication among semantic web agents. We propose architecture, as depicted at Figure 3.4 and we are inspired by the work in [90], for the server and its components and their interaction. The figure shows the capability for inter-communication amongst servers at different NetMem levels to exchange data/semantics based on needs of servers' NMemAgents. Also, Internet elements in different network domains can communicate with NetMem servers requesting specific data/semantics classes. The Application Programming Interface (API) component, e.g., HTTP plug-in, will forward HTTP requests/responses for data or concepts from/to Internet elements (e.g., users and applications) to/from NMemAgent. The server maintains two tables: a) criteria profile table; and b) update table. Those tables help the comprised NMemAgent to update already maintained data/semantics and to meet requests of data/semantics sent

by network entities and/or neighboring NMemAgents. The following subsections discuss criteria profile and update tables located in servers which host NMemAgents.

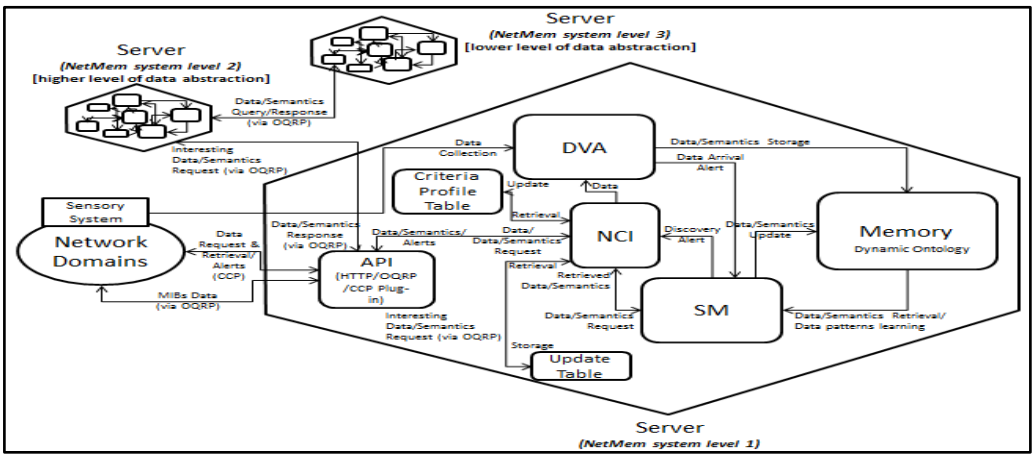


Figure 3.4 Server for hosting NMemAgent

Criteria Profile Table (CPT)

CPT is constructed to provide descriptions for multi-criteria that can be used to define bases on which requests/responses for various data/semantics will be executed. CPT maintains different criteria on which data/semantics can be retrieved like: a) location-based criteria, such as IP addresses; b) concern-based criteria like network communication concerns related to communication protocols, such as TCP protocol; c) entities' capability-based criteria like specific type of resources or sensors; and d) semantics-based criteria, such as the behavioral aspect of a concept class. CPT is found at each server and is used by NMemAgents to construct OQRP request/response messages to network entities/NMemAgents and CCP response messages to network entities. CPT is managed and updated by NCI at NMemAgents where the scale of CPTs depends on related NMemAgents experience. New criteria can be defined when agents receive update messages from other agents/entities regarding unfamiliar criteria, they will register those criteria and consider given inputs as defaults ones for those criteria. CPT fields as shown at Table 3.3:

1) ***Key***: it names or identifies the interesting criterion based on which data/semantics will be retrieval;
2) ***Inputs***: it shows criterion-related attributes which represent data/semantics features of interest; and
3) ***Description***: it interprets the relation/interdependence between criteria Keys and related inputs.

Table 3.3 Criteria Profile Table

Key	***Inputs***	***Description***
IP address	Communication protocol, type of resource (ToR), ……, etc.	Send available information about that IP address w.r.t given inputs (e.g., ***IP:ToR*** means send back knowledge about used resource at the network entity of IP). If there are no inputs, send all available information regarding that IP address
MAC address	Routing protocol (RP), ToR, data semantics such as concept_class: normal behavior of TCP protocol, ……, etc.	Send available information about that MAC address w.r.t given inputs. (e.g., ***MAC:RP*** → send information about the RP used at the network entity of MAC address)
TCP protocol	TCPHeaderFields,MaxSegmentSize,abnormalBehavior	Send what learned about TCP and given inputs
Storage memory (SM)	Storage Space, type of memory (ToM), ……, etc.	***SM:ToM*** → send information about this type of memory
⋮	⋮	⋮
Concept_ Class	Functional, behavioral, and structural (FBS) aspect	Send the FBS aspects of the required concept class.

Update Table (UT)

UT is constructed to keep up with requirements of update messages sent by NMemAgents requesting updates about various data/semantics, which are related to multi-criteria. UT maintains multiple entries where each entry represents an update message received by a NMemAgent from another NMemAgent/network entity concerning required data/semantics based on specific criteria. UT is managed and updated by NCI at NMemAgents where the scale of UTs depends on number of update messages received by those NMemAgents. The UT fields as shown in Table 3.4 are:

1) ***Source IP address***: it identifies the IP address of the update/query message source;

2) ***Msg code***: it represents the type of message sent by NMemAgents and it takes value "01" or "10" where "01" means update message and this entry will be available in UT even data/semantics are sent and "10" means query message and the entry will be dropped out when a query's response is generated;

3) ***MsgID***: it shows the identifier of the update message sent by the source where response messages regarding any updates will use that identifier;
4) ***Update Time Counter***: it gives the storage duration of each entry from the moment NMemAgents receive OQRP update/query messages from other NMemAgents or network entities; and
5) ***Criterion/Concept Class***: it describes the criterion/concept class and its related inputs based on which data/semantics will be retrieval

Table 3.4 Update Table

Source IP address	***Msg code***	***MsgID***	***Update Time Counter (time unit)***	***Criterion/Concept Class (key:inputs)***
192.168.32.15	01	56	200	File_Transfer_Service:Communication_Protocol
217.168.100.24	01	7	180	TCP:normal_behavior
217.168.100.24	01	7	130	0032.AB05.0000.07FF:*
200.123.10.66	10	33	100	217.168.100.24:type_of_memory
⋮	⋮	⋮	⋮	⋮

3.4.1.1 Inputs/outputs to/from NMemAgent and operation cases

In this subsection, we discuss the different operation cases (inputs/outputs) of NMemAgents as illustrated in Figure 3.5.

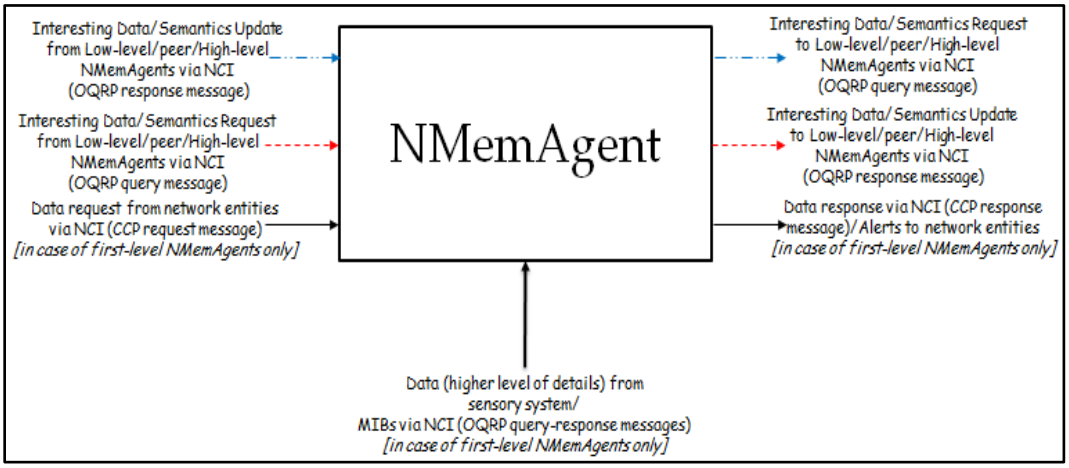


Figure 3.5 NMemAgent operations

- Data Collection Process (input process):
  - *Via Sensory systems* (e.g., sensor networks, networking tools for network traffic sniffing): NMemAgents, which are located at the lowest NetMem level, collect raw network-data (e.g., Internet traffic data) with high levels of details. Collected data are represented and stored by DVA component of each NMemAgent in their storage memories. SM will reason about data semantics with higher levels of details and express those semantics as concept classes clarifying their functional, behavioral and structural aspects. NMemAgents will send extracted semantics, to the connected high-level NMemAgents (agents at higher NetMem levels) via their NCI adopting OQRP update message.
  - *Via Management Information Base (MIB):* NMemAgents can collect management information from network entities' MIBs, which have a defined ontology of those entities' objects or attributes. Each object is identified by unique identifier and it refers to characteristics of related entities. NMemAgents utilize via NCI the OQRP to communicate with entities and get raw data about their objects, which are related to various Internet elements (e.g., services and protocols). For instance, TCP-based network entity will have a TCP object at its MIB which can give information about TCP statistics and behavior. MIBs' data will be represented and stored by DVA at NMemAgents' memories where SM will learn data patterns and extract semantics.
  - *Via NetMem agents (or NMemAgents):* NMemAgents will get data/semantics with lower and higher levels of abstraction and updates about data/semantics of interest from other NMemAgents located at the same NetMem level (same level of data abstraction), the next high level (agents have higher level of data abstraction) or the underlying low level (agents have lower level of data abstraction). Agents will get these data/semantics updates based on certain criteria and using the OQRP via NCI. Interesting data/semantics request messages can be sent from NMemAgents to other agents. SM will perform a matching process between required data/semantics and already maintained data/semantics in agents' memories. Once SM finds similarity, it sends NCI the results of data/semantics discovery process. On the other hand, if SM does not find the required data/semantics, it will send NCI a discovery alert declaring that data/semantics are unavailable. Accordingly, NCI will send other NMemAgents requesting data/semantics.

- Data/Semantics Request Process (input process):
  - *Data request from network entities:* NMemAgents at the lowest NetMem level are capable of handling data request messages from network entities in various network domains using the CCP protocol. Entities can get information about various Internet elements (e.g., applications and protocols) specifying certain

interesting aspects if needed. NMemAgents will receive those messages via NCI which will send them to SM to get required information. If SM is not able to meet messages' demands (i.e., required data), SM will send NCI to forward data semantics request message to upper NMemAgents or neighboring agents in the same level using the OQRP as discussed in the next point.

- *Interesting Data/Semantics Request from Low-level/peer/High-level NMemAgents:* NMemAgents at a NetMem level might receive interesting data/semantics OQRP query messages from agents at the same level, the next high level or the underlying low level. Requesting agents will be interesting in specific data/semantics based on mentioned interest criteria in query messages. Receiving NMemAgents will search in their local storage memories about required data/semantics. Those agents will send updates about interesting data once they have changes about this data in their memories. Those agents might learn required data/semantics before; however, they are now unavailable because they were dropped out agents' memories. Also, those agents might not learn this data/semantics before. In case of last two cases, NMemAgents will send other NMemAgents OQRP query messages via NCI requesting unavailable data/semantics.
- *Interesting Data/Semantics Update from Low-level/peer/High-level NMemAgents:* NMemAgents located at a NetMem level receive OQRP response messages via NCI based on interesting data/semantics OQRP query messages sent to other NMemAgents located at the same level, next high-level or underlying level. Responding NMemAgents send available data/semantics based on specified interests criteria at query messages. Those agents will send updates to requesting NMemAgents based on any changes they have for required data/semantics.

- Data/Semantics Response/Alerting Process (output process):
  - *Interesting Data/Semantics Request to Low-level/peer/High-level NMemAgents:* NMemAgents at a NetMem level can send interesting data/semantics OQRP query messages via NCI to NMemAgents at the same level, the next high level or the underlying low level. Requesting NMemAgents will specify criteria of data/semantics of interest like determining for example: a) concern-based information via specifying information related to certain network concerns (application, communication, and resource); and b) location-based information via specifying locations where data/semantics of interest will be existed. If there is matching between defined criteria and data/semantics in receiving agents, they will send retrieved data/semantics to requesting agents.
  - *Interesting Data/Semantics Update to Low-level/peer/High-level NMemAgents:* NMemAgents at a NetMem level send available interesting data/semantics using OQRP response messages via NCI to agents in the same level, the next high level

or the underlying low level. Those NMemAgents search for required data/semantics based on defined criteria like naming specific network concerns.

- *Data response/alerts to network entities:* NMemAgents in the lowest NetMem level respond to CCP request messages sent by network entities to learn data/ semantics with various levels of abstraction about various Internet elements. Those agents will send CCP response messages, which include data showing different data aspects, through NCI. Also, due to raw data collected via sensory system and/or MIBs and based on learned semantics, first-level agents can send network entities alerts which might refer, for example, to abnormal behavior of a service or a protocol.

3.5 NetMem Functions

In this subsection, we discuss NetMem functionalities executed by NMemAgent's comprised components, which are illustrated in Figure 3.2.

3.5.1 Data Virtualization and Access

This task is executed at DVA component. Figure 3.6 describes the proposed DVA component. DVA comprises the following units:

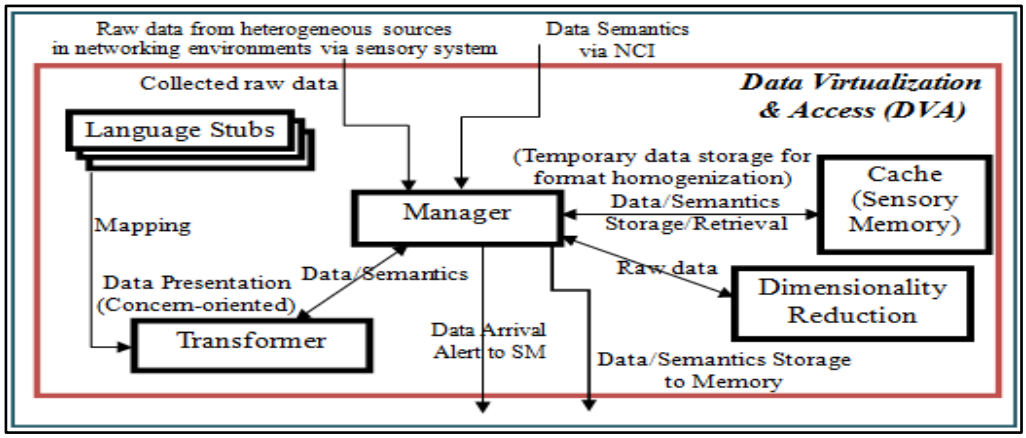


Figure 3.6 Data collection, virtualization and access in NMemAgents

a) Manager which handles raw data from sensory system and semantics via NCI component for data/semantics representation in storage memory's big data tables;

b) Cache which is the temporary warehouse where collected and received data/semantics are stored by the manager to facilitate learning formats of data/semantics, performing format mapping processes and dimensionality reduction tasks. In addition, the cache helps in enhance the response time of the DVA component since it can know the format of frequently collected data/semantics in a timely manner;

c) Dimensionality reduction is attached with algorithms for probabilistic dimensionality reduction (e.g., LSH [15]) of collected massive high-dimensional raw data;

d) Transformer which represents uniformly, using defined data models, data as profiles which comprise attributes or features categorized into groups based on application, communication and resource concerns; and

e) Language Stubs represent plug-in drivers that are used by the transformer to map from various representation formats to the format adopted by NetMem. For instance, DVA might collect raw data from three sources with different formats: a) MIBs at Internet TCP-based hosts; b) IPv4 Internet traffic packets of TCP-based services; and c) TCP's entries at Structured Query Language (SQL) server databases. DVA might adopt eXtensible Markup language (XML) to provide a virtual view of TCP object identifier's entries (e.g., maximum TCP retransmission timeout) and data attributes (e.g., the used "protocol" and "packet size") found in MIBs, IPv4 packet headers, and entries at SQL databases.

Due to complexity and scale of operating computer networks and Internet, tons of high and multi-dimensional network-data are generated from various sources. DVA will gather data via a sensory system (e.g., sensor networks with generic or specific type of sensors). DVA will adopt data models to represent uniformly in the storage memory data from heterogeneous sources as profiles of attribute-value pairs normalizing range of the same type of attributes. DVA uses a data model defined by eXtensible Markup Language (XML) [89] as a representation language clarifying type and attributes per each data profile. DVA adopts dimensionality reduction techniques for reducing data dimensionality. The following subsection discusses locality sensitive hashing (LSH) algorithm [15] which is adopted by DVA for data dimensionality reduction.

3.5.2 Data Dimensionality Reduction

Network-data characteristics include massive volume and high-dimensionality. Dimensionality reduction algorithms in NetMem can aid in improving its performance. There are various dimensionality reduction algorithms that can be utilized by NetMem, such as LSH [15], linear discriminant analysis [31] and principal component analysis [32]. The DVA employs LSH for reducing data dimensionality and finding similarities amongst huge amounts of high- and multi-dimensional data due to its following desirable features:

- Supporting high-dimensional network-data by hashing and detecting approximate neighboring points;
- Supporting multi-dimensional network-data via the capability of applying directed attribute or feature selection processes through designing multi-hash functions for certain set of attributes or features that are of interest and related to different ACR concerns. For example, a specific group of attributes related to interesting TCP-based file transfer service can be selected to be used in representing data profiles of such service;
- Random sampling process for data based on hashing can enhance time complexity of

DVA's functions; and

- Similarity search process leads to efficient data storage based on determining similarities.

The usage of LSH for data dimensionality reduction would result in loss of some information which might affect NetMem operation. The degree of data loss depends on the value of hashing function length and the attributes' group from which attributes are selected to form reduced dimensionality data profiles. LSH can be used in a form of directed attributes' selection. In other words, it is possible to define group of interesting attributes from which LSH will select to form hash functions. This can enhance information loss rates due to dimensionality reduction processes performed by LSH.

NMemAgents utilizes LSH with bit sampling for hamming distance where LSH selects attributes using hash functions which might be chosen randomly or according to specific directions like focusing on certain network concerns. Selected attributes will be used to construct reduced-dimensional data profiles in agents' storage memories. LSH helps in annotating data and finding similarities amongst data thus aiding in minimizing storage space for data recollection and patterns learning. Accordingly, this would aid in reducing the time complexity of NetMem. The pseudo code is shown in Figure 3.7.

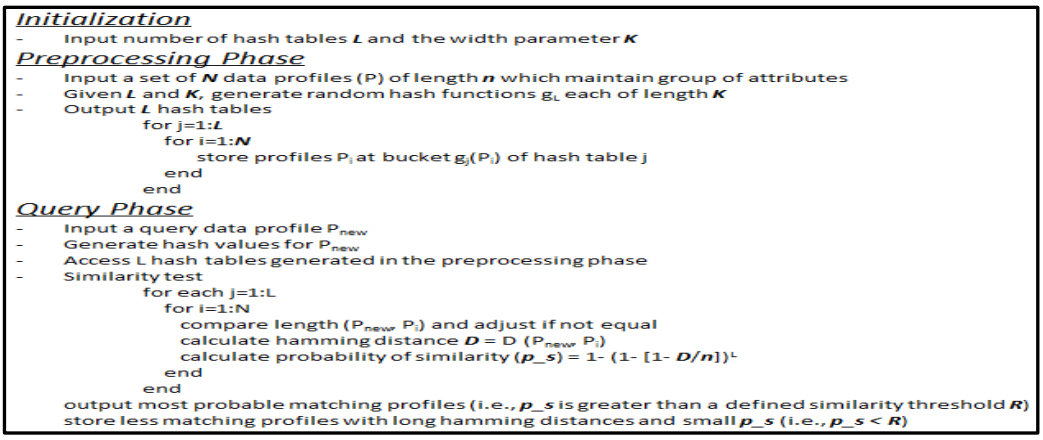


Figure 3.7 LSH Algorithm executed in DVA

The operation of LSH depends on identifying the number of hash functions and the length of hash functions (i.e., number of attributes after reduction). For each defined hash function there is a hash table that maintains hash values computed for stored reduced-dimensional profiles based on the related hash function. For finding similarities among data profiles and deciding storage in the memory, LSH builds hash tables where similar data profiles that have matching hash values (i.e., zero hamming distance) with specific hash functions will be maintained in the same place in related hash tables. Maintaining new collected reduced-dimensional data profiles in the storage memory will depend on calculating their hash values according to defined hash functions and comparing those values with already calculated and stored ones in hash tables (i.e., measuring the

hamming distances). Then, a similarity degree for each new profile is calculated based on their measured long hamming distances. The storage process for new data profiles in memory's big data tables will be done if the similarity degree of each profile is below a defined threshold of similarity. The following paragraph summarizes LSH operation within NetMem for data dimensionality reduction and finding similarities. LSH operation within the DVA involves the following processes:

1- Get new high-dimensional data profiles of length ***n***.
2- Build ***L*** hash tables depending on calculating hash values for already kept profiles in the storage memory adopting ***L*** defined hash functions (those functions refer to set of interesting data features or attributes where each hash function is of length ***k*** where ***n***>***k***).
3- Calculate ***L*** hash values for each new ***n***-dimensional data profile.
4- Investigate ***k***-length hash values with long hamming distance for each new data profile by comparing its computed ***k***-length hash values with the ones of already stored profiles.
5- Calculate the similarity degree for each new ***n***-dimensional data profile using its ***k***-length hash value with the longest hamming distance.
6- Generate new ***k***-dimensional data profiles which possess ***k***-length hash values achieving small similarity degree compared with a defined similarity threshold.
7- Store new ***k***-dimensional data profiles.

Example

NetMem collects raw data regarding TCP-based service from different sources (e.g., Internet traffic and TCP hosts' MIBs) through *T* time units to identify normal and abnormal behavior of related services. Raw data are represented by DVA using data models as profiles of attribute-value pairs. For example, a captured profile might have multiple instances with the following eleven attribute-value pairs: P = {date:value, time:value2, src_IP:value3, dest_IP:value4, packet_size:value5, packet_type:value6, service:value7, protocol:value8, port_number:value9, Seq_No:value10, MAC_add:value11}. Without using data dimensionality reduction techniques, such as LSH, different and various instances of data profiles constructed by DVA consume large space (e.g., hundreds Gigabytes) of storage memory. So, DVA adopts LSH with bit sampling for hamming distance to reduce the dimensionality of each profile and to find similarities amongst new captured profiles and already maintained ones targeting storage space minimization (e.g., tens of Gigabytes). We define for LSH six directed hash functions, each of length four, that maintain certain interesting attributes. Those attributes are (packet_size, packet_type, service, protocol, port_number). A reduced dimensional profile P_LSH after applying LSH over P might be as follows: P_LSH = {packet_size:value5, packet_type:value6, service:value7, port_number:value9}.

3.5.3 Semantics Reasoning

This process is executed at semantics manager (SM) component of each NMemAgent in NetMem. SM uses associative rule learning [21] to recognize attributes or features of data profiles maintained by DVA in the storage memory. SM utilizes Fuzzy membership functions (FMF) [22] to classify extracted attributes. The classification process for attributes depends on their related symbolic and numeric values. Monolithic and hybrid intelligence techniques (HIT) can be used to design reasoning models for NMemAgents to manage network-semantics and build DNCO. The capability (e.g., latent features extraction ability, high prediction accuracy, etc.) of adopted reasoning models for learning rich semantics depends on the operation performance of the used intelligence techniques. Various algorithms(e.g., LDA [23] or HMM [24]) can be utilized in SM for designing reasoning models. Figure 3.8 illustrates the architecture of SM in NMemAgents showing its main components.

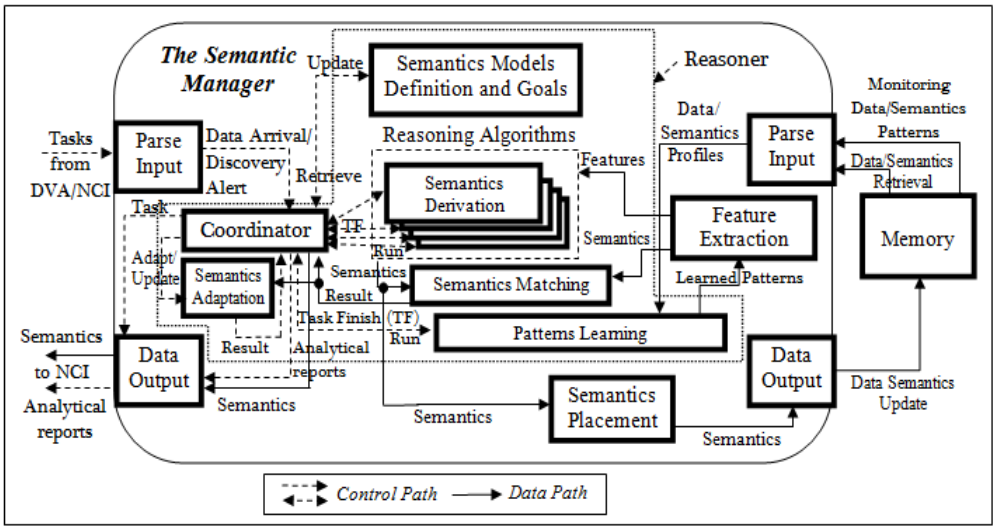


Figure 3.8 The semantic manager in NMemAgents

SM is responsible for generating/storing/updating semantics with higher levels of abstraction and retrieving semantics from the storage memory. SM operations are based on: 1) continual monitoring and learning for patterns of data/semantics with lower levels of abstraction; and 2) utilized statistical and probabilistic algorithms for reasoning and behavior estimation. SM uses machine learning algorithms like associative rule learning (ARL) [21, 91] to recognize features of data profiles maintained in the storage memory. Moreover, SM utilizes Fuzzy membership functions [22] to classify extracted attributes. Based on learned and classified attributes, SM adopts extraction and classification algorithms for learning high level latent features. SM runs models for semantics reasoning and matching. For instance, SM might adopt monolithic intelligence techniques for semantics reasoning like implementing HMM-based reasoning model. HMM [24] is a categorical sequence labeling supervised/unsupervised algorithm (predicting sequences of categorical labels). Sequence labeling can be treated as a set of independent classification tasks. Based on extracted features according to maintained data profiles in

the storage memory, SM runs HMM-based reasoning models after executing the training phase. The input to HMM-based models are sequence of profiles' features and the output is semantics (i.e., sequences of concepts) with certain probabilities. SM can use DTD XML language [92] as described later to represent extracted semantics which will be kept as associated concept classes. As shown in Figure 3.8, SM has the following components:

a) Coordinator which is responsible for handling and differentiating between enforcement signals from internal units, NCI and DVA to allow performing semantics derivation, retrieval, matching and discovery processes and to let pass networking data and semantics from the storage memory to SM;
b) Patterns learning which executes algorithms (e.g., associative rule learning [84]) and Fuzzy logic (e.g., Fuzzy membership functions (FMF) [22]) to learn data patterns in the storage memory. ARL can recognize attributes in maintained data profiles in the memory. FMF can be used to classify extracted attributes. Also, those pattern learning techniques help in learning semantics patterns, which describe relation or interdependence among different semantics, used in semantic matching process;
c) Reasoning algorithms which represent the semantics reasoning models via implementing group of AI algorithms, e.g., HMM algorithms, for multi-stage reasoning to perform tasks of semantics derivation at different levels of granularity, i.e., providing FBS data, and generation of semantics patterns;
d) Semantics matching which is embedded with behavior estimation models and using statistical analysis algorithms for applying semantics fitting and updating processes with already maintained semantics in the storage memory;
e) Semantics adaptation maintains algorithms for defining aspects and features of semantics considering changes happened in those semantics FBS data based on results from semantics matching unit and a control signal from coordinator;
f) Semantics definition and goals which shows experience of the SM, preserves semantic derivation models and definitions and intentions of semantics;
g) Semantics placement which has data storage models and definition languages, such as eXtensible Markup Language (XML) document type definition [92], for maintaining semantics in storage memories' big data tables;
h) Feature extraction and classification which maintains algorithms for probabilistic and statistical analysis and classification, such as Latent Dirichlet Allocation (LDA) [23], decision trees [93] and Fuzzy membership functions [22];
i) Parse input which is responsible to handle inputs to SM whether those inputs are data/semantics from the storage memory or alerts and request signals from DVA and NCI, respectively; and
j) Data output which is the gate where analytical reports and data/semantics are passed through to NCI and storage memory/NCI, respectively.

Each NMemAgent in NetMem stores data patterns which can be utilized to recognize network-semantics. Each NMemAgent will maintain recognized semantics as associative network concepts at different levels of granularity. Accordingly, NetMem will form dynamic ontology of network concepts, which are represented by the FBS (Function-Behavior-Structure) engineering design framework [20]. This enables the identification of network concepts through providing their functional, behavioral and structural aspects. For example, NetMem can learn data semantics concerning TCP and UDP protocols (i.e., communication concerns) within file transfer services (i.e., application concern) in wireless contexts (i.e., a communication concern). Those semantics can be used to construct a conceptual model, which describe: a) functions (i.e., functional aspect) of TCP and UDP protocols; b) the normal and abnormal behavior (i.e., behavioral aspect) of those protocols and the different behavior classes that can emerge (e.g., different attacks under the abnormal behavior class); c) relations (i.e., structural aspect) between concept classes (e.g., overlapping TCP abnormal behavior classes that share common features).

A concepts ontology is formed by NetMem and updated over time showing various and correlated classes of concepts [94]. For example, NetMem can learn patterns to recognize the normal and abnormal behavior of widely used communication protocols, such as TCP. TCP patterns yield some information related to various network concerns [5]. For instance, group of Internet services and application that operate on top of TCP can be determined. Furthermore, the normal physical parameters like TCP session idle timeout, TCP packet size, and maximum congestion window size within wired communication contexts can be known. Also, normal ratios of sent-to-received TCP packet and ratios of TCP SYN sent packets with respect to TCP data packets can be identified. Hence, studying TCP patterns makes NetMem reason about semantics that are used in building TCP concepts (e.g., TCP behavior models) which describe TCP normal and abnormal behavior. Those TCP concepts can be used by NetMem to match behavior of unknown communication protocols that exhibit behavior like TCP's behavior. Moreover, retrieving TCP concepts enable discovery of associated TCP information related to various network concerns and FBS aspects. For example, TCP concepts can show the behavior of TCP and normal settings of its parameters in wireless communication environments.

3.5.4 Data/Semantics Collection and Representation

Due to the complexity and scale of operating computer networks and the Internet, tons of data, with high dimension of attributes and different representation models, are generated. DVA component in NMemAgents will gather data/semantics according to the following two cases:

a) DVA of NMemAgents in the lowest NetMem level will collect raw data from heterogeneous systems with various formats to make an abstract view of data. DVA might collect raw data, via adopting network management protocols and/or sensory systems using remote terminal units [66], from various sources with different formats,

such as a) Management Information Bases (MIB) [18]; b) Internet traffic packets; and c) entries at Structured Query Language (SQL) database servers.

b) DVA of NMemAgents at upper NetMem levels will collect data/semantics from agents at lower levels. Data/semantics will be represented by known data representation models adopted by DVA and SM in NMemAgents.

DVA represents raw network data and semantics uniformly in the storage memory which comprises relational big data tables. Data/semantics representation by DVA enables and facilitates: i) data/semantics patterns learning, abstraction and retrieval processes, and ii) updating already kept data/semantics in the memory. In case of raw data (i.e., NMemAgents in the lowest level), DVA maintains raw data in service data structured tables (SDST)) where they represent data as profiles of attribute-value pairs where each attribute can be classified based on three main network ACR concerns as defined in [5]. DVA might adopt XML language to provide homogeneous representation of data collected from various sources. This will facilitate data access and management by diverse Internet elements through stating the related three network concerns' attributes. In case of extracted semantics (i.e., NMemAgents at upper levels) DVA will be keep recognized semantics in the storage memory into relational big tables, namely, class and sub-class tables. Semantics in class tables are represented as entries of concept classes with features. Concept classes are classified according to ACR concern showing their FBS features. Concept classes with common features related to various ACR concerns will have associations which will be clarified in sub-class tables. SM will able to update data/semantics entries in SDST and class and sub-class tables.

For example, DVA uses XML language to provide a virtual view of TCP object identifier's entries (e.g., maximum TCP retransmission timeout) and data attributes (e.g., the used "protocol" and "packet size") found in MIBs, IPv4 packet headers, and entries at SQL databases. Figure 3.9 shows an example of data mapping process from SQL to XML for ACR concern-oriented representation in the storage memory of NMemAgents. Figure 3.10 and Figure 3.11 provide examples of XML code for representing raw data and

SQL maintains data in tables

```
CREATE TABLE network_services
(service_name char(30) not null,
protocol_name char(20) not null,
routing_name char(20) not null,
packet_type char(20) not null,
packet_size int(15) not null,
mac char(15) not null,
encryption char(15) not null)

INSERT INTO network_services
values('Video_Streaming', 'TCP', 'CA-AODV', 'SYN', '128', 'IEEE 802.11a', 'DES')
```

Service	*Protocol*	*Routing*	*Packet Type*	*Packet Size*	*MAC*	*Encryption*
File_Transfer	UDP	AODV	Data	1024	IEEE 802.3	DES
Video_Streaming	TCP	CA-AODV	SYN	128	IEEE 802.11a	DES
----	----	----	----	----	----	----

Mapping

```
SQL: ''network_serivces'' => XML: ''DataEntry''
SQL: ''Service'' => XML: ''Application_concern'', SQL: 'File_Transfer' => XML: 'Data_File_Transfer'
SQL: ''Protocol''=> XML: ''Communication_concern'', SQL: 'UDP' => XML: 'UDP_communication_protocol'
SQL: ''Routing''=> XML: ''Communication_concern'', SQL: 'AODV' => XML: 'AODV_reactive_routing_protocol'
SQL: ''Packet Type''=> XML: ''Application_concern'', SQL: 'Data' => XML: 'data_packet'
SQL: ''Packet Size''=> XML: ''Application_concern'', SQL: '1024' => XML: 'size:1024'
SQL: ''MAC''=> XML: ''Resource_concern'', SQL: 'IEEE 802.3' => XML: 'mac:IEEE:802.3'
SQL: ''Encryption''=> XML: ''Resource_concern'', SQL: 'DES' => XML: 'DES_encryption'
```

Figure 3.9 Mapping process by DVA from SQL database to XML

semantics, respectively, in storage memories of NMemAgents. DVA of NMemAgents at upper NetMem levels can read and define entries for data semantics with FBS feature-value pairs related to collected concept classes in NMemAgents at lower NetMem levels.

```
<?xml version="1.0" encoding="UTF-8"?"
<DataEntry>
    <Service ="File_Transfer" RecordedBy="DVA" WhenRecorded="20130719">
    <ServiceDefintion>
        < SDST_TableID >ID</SDST_TableID>
        <PrinicpleName> File_Transfer </PrincipleName>
    </ServiceDefintion>
    <NetworkConcernRef xlink:href="#id('Domain1')">Application</NetworkConcernRef>
    <NetworkConcernRef xlink:href="#id('Domain2')">Data Transfer</NetworkConcernRef>
<ApplicationConcern>
    < Attributes xlink:href="Application
    Attributes.xml#id('Attributes')">Data_File_Transfer,Data_Packet,Size:1024</Attributes>
</ApplicationConcern >
<CommunicationConcern >
    < Attributes xlink:href="Communication  Attributes.xml#id('Attributes')">UDP_communication_protocol,
    AODV_reactive_routing_protocol </Attributes>
</CommunicationConcern >
<ResourceConcern>
    < Attributes xlink:href="Resource Attributes.xml#id('Attributes')"> mac:IEEE:802.3 , DES_encryption
    </Attributes>
</ResourceConcern>
</DataEntry>
```

Figure 3.10 XML code for a data entry in the memory of a NMemAgent at the lowest NetMem level

```
<?xml version="1.0" encoding="UTF-8"?"
<ConceptClassEntry>
    < ClassType ="Application" RecordedBy="DVA" WhenRecorded="20130719 "UpdatedBy="SM" WhenUpdated="20130719">
    < Class_TableID >ID</Class_TableID>
    <ClassDefintion>
        <ClassID> TCP_based_File_Transfer </ClassID>
        <ClassAssociation>
            <ConceptClassRef xlink:href="#id('SubClassTableID')">Service</ConceptClassRef>
            <ConceptClassRef xlink:href="#id('SubClassTableID')">File_Transfer_Service</ConceptClassRef>
        </ClassAssociation>
    </ClassDefintion>
    <ClassFeatures>
        <FunctionalAspects>
            <Aspects xlink:href="Functional Features.xml#id('Aspects')">Function:Reliable_Data_File_Transfer</Aspects>
        </FunctionalAspects>
        <BehavioralAspects>
            < Aspects xlink:href="Behavioral Features.xml#id('Aspects')">MaxCongestionWindowSize:value,
            TimeToIdle:value,MaxRetransmissionTimer:value,NormalDataToSynPacketsRatio:value,
            NormalSentReceivedPacketsRatio:value</Aspects >
        </BehavioralAspects>
        <StructuralAspects>
            <Aspects xlink:href="Structural Features.xml#id('Aspects')"> application: medium_packet_size,
            encryption_DES; communication:TCP_protocol, AODV_routing_protocol ; resource: mac_IEEE:802.3 </Aspects>
        </StructuralAspects>
    </ClassFeatures>
</ConceptClassEntry>
```

Figure 3.11 XML code for a semantics entry in the memory of a NMemAgent

3.5.5 Data Attributes Discovery and Classification

Once, DVA accomplishes its tasks of data/semantics representation in the storage memory, it sends an alert message to SM. SM begins to learn patterns of data/semantics stored in the memory. SM runs ARL algorithms (as shown in Figure 3.12) and FMF to learn unique profiles and groups of attributes and classify those attributes based on their values, respectively. Implemented ARL algorithms will able to discover sets of unique attributes in all data profiles stored in the storage memory. Each attribute is associated with a value which might be numeric or string. Defined FMF can be used to classify each attribute depending on its value as illustrated in Figure 3.13. For instance and through a defined reasoning window, SM aggregates information, initially, via learning patterns of TCP data profiles through recognizing the frequency of each data profile in the storage memory and extracting and classifying data profiles' comprised attributes. For example, Analyzing TCP data profiles in NMemAgent's storage memory by ARL and FMF might

give the following classified attributes or features: 10000 profile's instances, large_TCP_packet_size, TCP-SYN_packets, file_trnasfer_service.

```
N_A: average number of attributes per data profile
P: data profile which contains attributes
E_A: extracted attributes
Define min_support; define the minimum number of data profiles that have required attributes
                    (based on number of captured data profiles in the storage memory)
For (n=2; n ≤ N_A; n++) {
        initialize count; counter for calculating number of data profiles' instances
        Generate candidate group C_A of n attributes; generated by SM
        For data profiles in the storage memory do {
                For all attributes in each P do {
                        if (comprised attributes found in C_A) count++;
                }
        }
        if (count ≥ min_support)  E_A = C_A
}
Output: E_A; a final set of extracted attributes
```

Figure 3.12 Pseudo code of the used associative rule learning with apriori algorithm

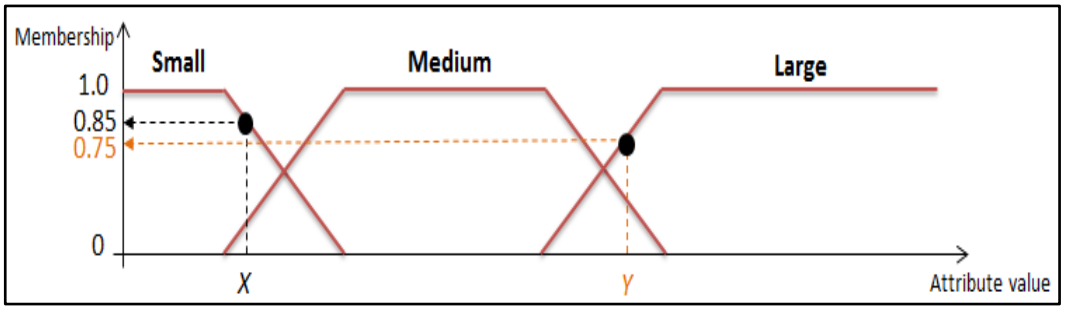


Figure 3.13 Example of a Fuzzy membership function

3.5.6 Cloud-like Data Storage and Storage Memory in NMemAgents

NetMem adopts cloud-like data storage techniques [17, 80], such as extensible big relational tables [25] to maintain high- and multi-dimensional large-scale data and extracted semantics. DVA component in each NMemAgent collects raw data and data/semantics with lower levels of abstraction and unifies data/semantics representation in the storage memory. Represented data/semantics in the storage memory include attributes or features related various application, communication and resource network concerns as defined in [5]. On the other hand, SM component in each NMemAgent learn patterns of maintained data/semantics to reason about semantics with higher levels of abstraction. Extracted semantics will be maintained in storage memories of agents at high NetMem levels. The following two subsections represent two kinds of storage memories which are constructed by relational big data tables to maintain raw data and semantics with higher levels of abstraction. The first kind is for storage memories which keep raw data and are located in NMemAgents at the lowest NetMem level. The other kind is for

storage memories that maintain semantics and are located in NMemAgents at NetMem level above the lowest level.

3.5.6.1 NetMemAgent Memory at the lowest NetMem level

The storage memory of each NMemAgent at the lowest NetMem level stores temporarily raw data of different Internet elements (e.g., services and applications) enabling data patterns learning for semantics reasoning via SM. SM will reason about semantics with higher levels of abstraction and send semantics to agents in the next upper NetMem level. Communicating entities can access and learn at runtime and on-demand maintained data in memories of NMemAgents in the lowest NetMem level to enhance their networking operations. In addition, entities can request semantics from agents in the lowest NetMem level. SM in those agents can try to get required semantics based on kept data. In case of SM inability, SM will send semantics request to agents in same or upper NetMem levels.

Figure 3.14 shows the structure of storage memory within NMemAgents resided in the lowest NetMem level. We propose five dataset tables, namely service data structured table (SDST), concerns data structured table (CDST), concerns index table (CIT), composite concerns index table (CCIT) and concept class and sub-class tables. SDST contains entries for services and their related networking concerns. CDST extracts each concern's data entry service in the SDST where it defines FBS attributes per each concern. The CIT and CCIT clarify in which entries in the SDST, a single concern's and group of concerns' attributes can be found, respectively. Concept class and sub-class tables contain semantics which are represented as concept classes that are learned based on learning patterns of data maintained at the last four discussed tables. Concepts are maintained according to three network concerns. Those tables are found in memories of NMemAgents at upper NetMem levels. Extracted concepts at upper levels depend on learning patterns of semantics or concept classes registered in class and sub-class tables at agents at lower levels.

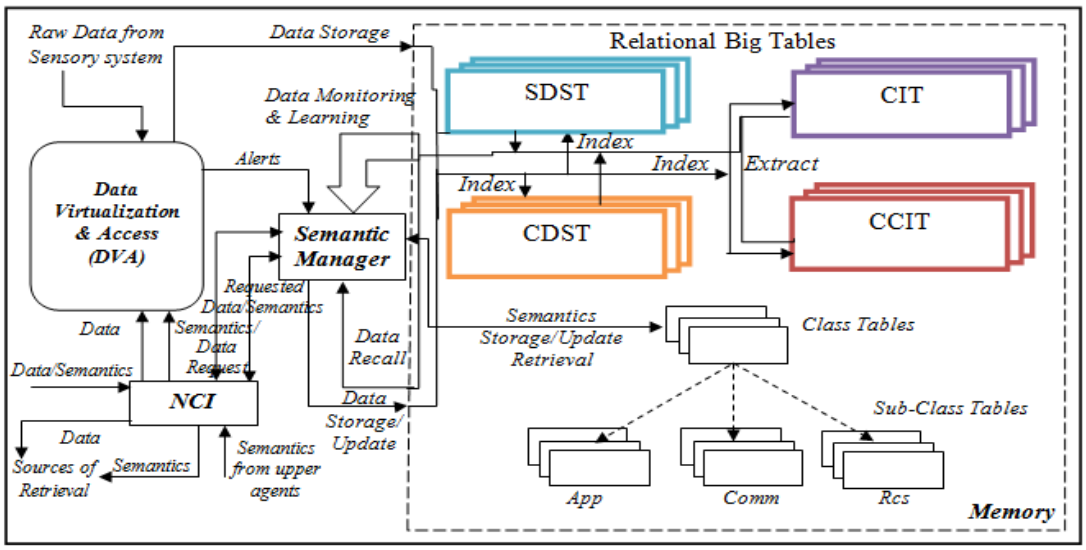


Figure 3.14 Memory structure in NMemAgents at the lowest NetMem level

Data Tables of memories in NMemAgents:

1) *Service Data Structured Table (SDST):* raw data are recorded by DVA component in that table (Table 3.5). This table consists of five column families (cf). The first one is date/time stamp entry where it reveals data and time of service occurrence. Second cf is the key which is a unique identifier for each record. The third cf is the supported service where that cf has two sub columns that one is for service's ID and the other is for service data raw. For unknown services, a label "unknown" is used. The next cf is for concerns' raw data (application, communication, and resource). Each communicating entity will be able autonomously to infer configuration process to extract concerns' raw data depending on each service type it supports. The last cf is for semantics graph where it provides the relation between different concerns' data in a service with other provided services in other SDST's entries. This helps in aiding communicating entities to calculate correlations between services and to provide unknown services depending on similar semantics. An extraction mechanism is required to extract concerns' data raw from the overall service raw. That mechanism can be offered by DVA, which has pre built-in tools to extract and present raw semantics' data of running services. Communicating entities can define new service types in case that no previous entries found at the SDST (i.e. unfamiliar services for those entities).

Table 3.5 SDST Table

Key	***DT***	***Service***		***Attributes***			***Semantic Gph***		
		ID	*Raw Data*	*Ap*	*Cm*	*Rs*	*Ap*	*Cm*	*Rs*
1	04/21 pm	S5	FileTransfer_rate 20….	FileTransfer	UDP_AODV.. ...	freq_20Mbps_DES ..	S3	S6	S3
2	04/22 pm	S3	FileTransfer_RTP_pkt_size_1024_freq_20Mbp	FileTransfer	RTP_CA-AODV_ ...	freq_20Mbps_DES –...	S5	S4	S5

2) *Concerns Data Structured Table (CDST):* This table (Table 3.6) presents attributes of each networking concern registered in the SDST by the FBS engineering framework [19, 20]. In another meaning, registered data per concern in CDST will be classified into functional, behavioral, and structural data. CDST table is extensible and dynamic in its structure depending on the number of extracted FBS data per concern. CDST consists of four column families. The first column refers to the entry key which is similar to that in the SDST. The second cf is for FBS data of the application concerns.

The third and fourth cf are for FBS of communication and resource concerns. For extracting FBS data from the networking concerns' data in SDST, there is a need for inferring a configuration setup using the built-in tools by DVA. The configuration performs FBS data formulation and synthesis over attributes of concerns in SDST. Also, attribute extraction mechanisms will be used where those mechanisms specify application, communication, and resource attributes.

Table 3.6 CDST Table

Key	*Ap Attributes*						*Cm Attributes*						*Rs Attributes*					
	ACn_1			*ACn_2*			*CCn_1*			*CCn_2*			*RCn_1*			*RCn_2*		
1	F	B	S	F	B	S	F	B	S	F	B	S	F	B	S	F	B	S
2	F	B	S	F	B	S	F	B	S	F	B	S	F	B	S	F	B	S

As an example for FBS data in the CDST, we will take and application attribute from the first row in the SDST table shown in Table 3.5. We have FileTransfer application which will be recorded in CDST table as an application concern which will have three FBS data entries. First, its functional data are to transfer data files. Second, its behavioral data show for example expected data rate at congested networks. Finally, its structural data define required policies as data packet format.

3) *Concerns Index Table (CIT):* This table (Table 3.7) facilitates the query and discovery processes where some interesting extracted attributes from a specific concerns' data raw can be indexed. CIT is an extendible big table with a dynamic size. The index cf includes the required attribute. The value cf contains references as the keys "unique identifier" in the CDST or SDST which indicates where interesting attributes can be found. For instance, CIT may contain an entry for identifier of a routing protocol (e.g. AODV); and it refers to which entries in SDST that protocol exists.

Table 3.7 CIT Table

Value	*Index*
key_1	ACn
key_2	CCn
……	……

4) *Composite Concerns Index Table (CCIT):* This table (Table 3.8) facilitates the query and discovery processes where composite of interesting extracted attributes from different concerns' data raw can be indexed. CCIT is an extendible table with a

dynamic size. The index cf includes the required composite attribute. The value cf contains references as the keys "unique identifier" in the CDST or SDST where required attributes occur. For example, an entry in the CCIT.

Table 3.8 CCIT Table

Value	*Index*
Key'_1	ACn_RCn
Key'_2	ACn_CCn_RCn
……	……

5) Concept Class and Sub-Class Tables:

a. *Class Table:* There are sets of extensible class tables (see Table 3.9) which are used to store and categorize extracted network-semantics as concept classes based on three network concerns (application, communication and resource) defined by three column families. Each class table has unique identifier. Networking concerns are kept as concepts classified to a certain concern based on output from SM reasoning models. For example, VideoStreaming is a network concern which is stored as an application concept class (i.e., a type of applications) and it is reserved in the first column under the category of application concepts with code {01}.

Table 3.9 Class Table

Table ID		
Concern ID {01}	*Concern ID {10}*	*Concern ID {11}*
A_1	C_1	R_1
A_2	C_2	R_2
……	……	……

b.Sub-Class Table

i) level-1 sub-class table

There are sets of extensible sub-class level-1 tables where each table has its own identifier (see Table 3.10). According to their identifier, sets of sub-class tables are divided into three sets; application, communication and resource sub-class tables. Inside sub-class tables, we have column families for concerns items which are represented in class tables (e.g. A_1 or C_1). The entries for each concern item are group of codes (CcnCode) which refers to column families (i.e. other concern item) in same/other sub-class tables. In another meaning, sub-class tables are relational big tables which show dependence among various networking concerns. CcnCodes are represented by numeric

sets where each set has generic format: {ConcernID, TableID, ColumnNumber}. For instance, CcnCode_1 of a concern item 1 in Table 3.10 may be {10, 2, 3} which tells that concern item 1 has relation with a communication concern (due to code 10) in another sub-class table with identifier 2 in the third column.

Table 3.10 Sub-class table (level-1)

Table ID			
Concern Item 1	***Concern Item 2***	***Concern Item 3***	……
CcnCode_1 CcnCode_2 CcnCode_3 ……	Ccn'Code_1 Ccn'Code_2 Ccn'Code_3 ……	Ccn"Code_1 Ccn"Code_2 Ccn"Code_3 ……	…… …… …… ……

ii) level-2 sub-class table

There are sets of extensible sub-class level-2 tables where each table is appended to a sub-class level-1 table (see Table 3.11). Sub-class level-2 tables maintain data casted to FBS engineering framework. In another meaning, each data of a concern item is abstracted into three sections; function, behavior and structure of that concern. So, data with more specific details about group of concerns can be extracted via those tables.

Table 3.11 Sub-class table (level-2)

Concern Item 1			***Concern Item 2***			***Concern Item 3***			……		
F	***B***	***S***	***F***	***B***	***S***	***F***	***B***	***S***	***F***	***B***	***S***

3.5.6.2 Memory in NMemAgents at NetMem levels above the lowest level

Memories in agents located in high NetMem levels are permanent auto-associative memories of valuable networking data (or semantics) with higher levels of abstraction. Semantics are stored as associated groups of concepts with different classes forming ontology of concepts. Semantics with higher levels of abstraction are defined based on learning patterns of data/semantics with lower levels of abstraction in low NetMem levels. Built concepts ontology in storage memories yields knowledge at different levels of granularity where concepts are classified according to application, communication, and resource concerns. Additionally, concept classes are represented showing their functional, structural and behavioral aspects using FBS engineering framework as defined in [19, 20].

Figure 3.15 shows storage memories' design in NMemAgents located in high NetMem levels. The design depends on using sets of extensible relational big tables which are called class and sub-class tables as discussed in the last subsection. Extracted semantics, represented as associated groups of concepts, comprise three classes of features registered

at class tables. Features per each class are stored at group of sub-class tables for that class. Entries, or constructed semantics, in class tables depend on categories of features found in sets of sub-class tables. Concepts are defined as group of correlated features. Features are represented via numeric vectors and alphabetic symbols. Class and sub-class tables and their contents build relational graph (or conceptual pattern) for all registered concepts. A conceptual pattern is a composition of concepts; and shows a description of inter-relationship between those concepts where their relationship might be containment or imply or dependence.

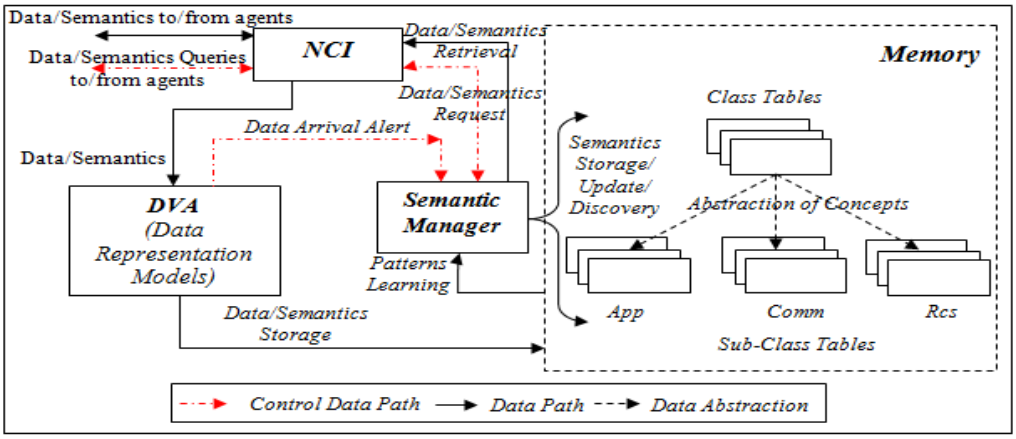


Figure 3.15 Memory structure in NMemAgents at NetMem levels above the lowest level

Figure 3.16 shows an example for class and sub-class tables. Firstly, we have sets of class tables which classify entries into three main categories (i.e. concepts): application, communication, and resource concerns. Each concept has its 2-ary digital code (e.g. application has code {01}) to be represented in sub-class tables. Class tables are extensible; and they contain multiple concepts. Each class table has unique identifier (ID). Then, we have sets of sub-class level-1 and level-2 tables. Sub-class level-1 tables are relational tables. Each sub-class level-1 table has also its unique ID and maintains codes for developing relations among entries and facilitating concepts' discovery and retrieval. Sub-class level-2 tables are appended tables to sub-class level-1 tables and their networking concerns' data. Each sub-class level-2 table contains networking concerns' data which are abstracted to functional-behavioral-structural data using FBS engineering [19, 20].

For example, in the first column of AP_1 in the sub-class level-1 table in Figure 3.16, we have the code {10,1,1} which is interpreted as follows: 10 is the 2-ary code for communication concept; 1 is the sub-class level-1 table ID of communication concepts; 1 is the number of column in the required sub-class level-1 table. So, AP_1 is an application which uses communication concept found in the first sub-class level-1 table in the first column. Via the used association learning algorithm, concepts {{10,1,1},{11,7,3},…..} will refer to AP_1. Hence, if a networking entity requests to retrieve concepts {10,1,1} and{11,7,3}, that entity is likely interesting in AP_1. For sub-

class level-2 tables, data of AP_1 are abstracted by SM into functional, behavioral, and structural data. Simply, the F column for AP_1 will maintain functions for that application as "video transfer with high quality". The B column will contain behavioral data, such as "maximum allowable latency in noisy mediums". The S column will have structural data as "video transfer policies".

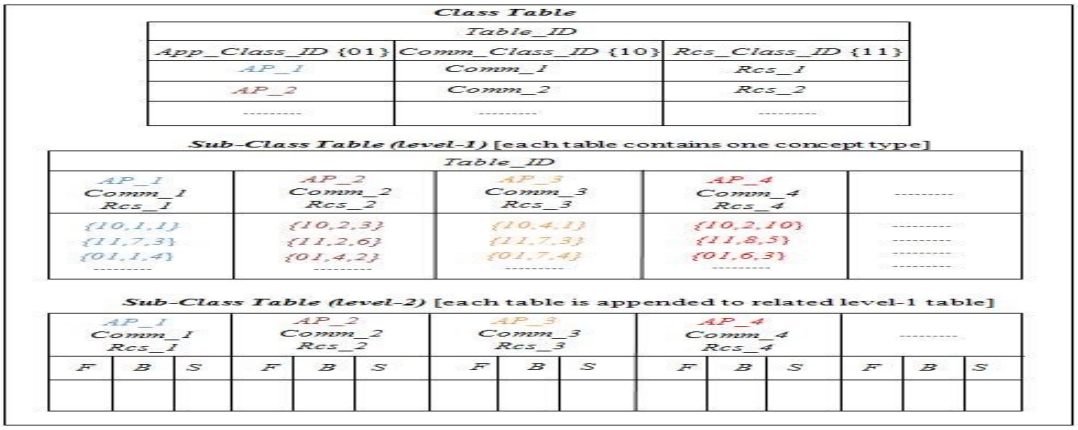


Figure 3.16 The proposed class and sub-Class tables for NMemAgents' storage memories

3.6 Dynamic Network-Concept Ontology (DNCO)

SM component of NMemAgents reason about semantics with higher levels of abstraction based on learning patterns of data/semantics with lower levels of abstraction maintained in NMemAgents at lower NetMem levels. SM utilizes semantics reasoning models that are able to reason about semantics or concept classes related to ACR concerns. Each extracted concept class will be registered based on the main concern it relates to and its recognized FBS aspects. Sets of combined reasoning models can be used by SM where a combined reasoning model comprises more than one reasoning model implemented by same or different reasoning algorithms. A combined reasoning model will extract concept classes related to various network concerns with capability of associations among known classes.

Each NMemAgent represents in its memory recognized semantics as correlated concept classes forming DNCO. DNCO is dynamic since agents have the ability to continually update DNCO entries based on ever-performing data collection process and discovering new semantics. Level of data abstraction in formed DNCO depends on the NetMem level where an agent localizes. The higher NetMem level is, the more enriched DNCO can be found in related NMemAgents. In other words, DNCO in upper NMemAgents has wider scope of and fine-granulated concept classes than DNCO in lower NMemAgents. Formed DNCO by NetMem possesses the following features: a) scalable growth of ontology comprising hierarchy of correlated concept classes at different levels of abstraction and granularity; and b) on-demand knowledge services for Internet elements to learn dynamic behavior models related to various elements. We

discuss in the following subsections levels of data abstraction in DNCO, how DNCO is built and updated and data/semantics retrieval form DNCO.

3.6.1 Levels of Abstraction

Recognized network-semantics are maintained as correlated concept classes classified based on ACR concerns and at different levels of granularity showing classes' related ACR concerns and FBS aspects. For instance, TCP- and UDP-based services exhibit patterns through data exchanged among networking hosts. SM learns the different data profiles, from multiple flows, and extracts their features. These features are then fed into the implemented reasoning models to extract data semantics concerning normal/abnormal behavior of file transfer services. SM classifies extracted semantics of file transfer services as concept classes under the application concern showing inter-related behavior concept classes in the storage memory. The association among registered concept classes under different categories related to various network concerns forms the DNCO. Lower level concept classes are child classes according to top level parent concept classes.

Many languages were presented to define ontologies and their contents. For example, Web Ontology Language (OWL) [95] was presented to represent semantic web and related ontologies. Also, eXtensible Markup Language (XML) [89] is used to represent data structures and ontologies (e.g., web services). A set of markup declarations that can provide for ontology languages (e.g., XML) building blocks to organize the content structure is presented by Document Type Definition (DTD) [92]. NetMem adopts DTD declarations, as illustrated in Figure 3.17, to provide a set of markup declarations that can provide for ontology languages (e.g., XML) building blocks to organize DNCO content structure as correlated concept classes showing their FBS aspects. Figure 3.18 shows an example DNCO that might be built in storage memory of a NMemAgent. As illustrated in the figure, there are various network concept classes related to different network concerns.

```
<!DOCTYPE ConceptClass[
<!ENTITY % Text "(div|p|ul|ol|table|h3|h4|h5|h6)*"  >
<!ENTITY % HTML-text-elements SYSTEM "XHTML.DTD" >
<!ENTITY % link-atts '
                xmlns:xlink     CDATA          #FIXED          "Link to access Storage Memory"
                xlink:type      (simple|extended|locator|arc)                  #FIXED "simple"
                xlink:role      CDATA          #IMPLIED
                xlink:title     CDATA          #IMPLIED
                xlink:show      (new|parsed|replace)          "replace"
                xlink:actuate   (user|auto)    "user" ' >
                <!ENTITY        % ISO8601Date                 "CDATA" >
                <!ENTITY        % metadata '
                                RecordedBy                    CDATA          #IMPLIED
                                WhenRecorded %ISO8601Date;  #IMPLIED   ' >
<!-- + quantifier means one or more, * quantifier means zero or more, ? quantifier means no more than one occurs, no quantifier means
that item must occur exactly one time>
<!ELEMENT ConceptClass(Concept+,TableID+) >
<!ELEMENT Concept ((ConceptTitle|PrincipleName)+, Definition, (BroaderConcept)*, (PartConcept)*, (ConceptType)+,
(ConceptRelationship)*,(ConceptFeatures)+) >
<!ATTLIST Concept               ID              #REQUIRED     %metadata; >
<!ELEMENT ConceptTitle      (PCDATA) >     <!ATTLIST Title xml:lang CDATA "EN" %metadata; >
<!ELEMENT Definition %Text; > <!ATTLIST Definition %metadata; >
<!ELEMENT BroaderConcept    (PCDATA) >    <!ATTLIST Title xml:lang CDATA "EN" %metadata; >
<!ELEMENT PartConcept       (PCDATA) >    <!ATTLIST Title xml:lang CDATA "EN" %metadata; >
<!ELEMENT ConceptType       (PCDATA) >     <!ATTLIST Title xml:lang CDATA "EN" %metadata; >
<!ELEMENT ConceptRelationship               (PCDATA) >      <!ATTLIST Title xml:lang CDATA "EN" %metadata; >
<!ELEMENT ConceptFeatures                   (PCDATA) >      <!ATTLIST Title xml:lang CDATA "EN" %metadata; >
%HTML-text-elements;        ]>
```

Figure 3.17 XML DTD language for defining DNCO structure with hierarchy of correlated concept classes

In DNCO, concept classes represent abstracted concepts or composition of more than one concept. An abstracted concept class can provide valuable information about a specific network concern, such as the normal behavior of TCP protocol. A composed concept class might give correlated information about more than one network concern, such as the abnormal behavior of UDP protocol with a delay sensitive application. That class combines a network communications concern "UDP protocol" with an application concern "delay sensitive application".

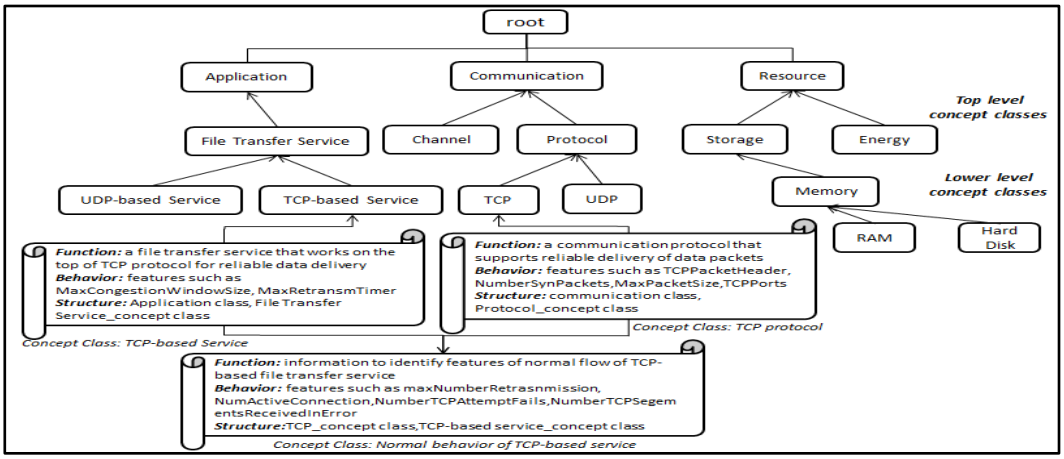


Figure 3.18 Example of dynamic network-concept ontology

3.6.2 DNCO Building and Update

SM builds DNCO and it continually updates the formed DNCO and the related hierarchical structure of concept classes. The process of building and updating DNCO in NMemAgnet comprises the following:

- -Discovery of and classifying attributes of raw data and semantics with lower levels of abstraction.
- -Extracting ***F*** latent high-level features based on recognized attributes and related to FBS aspects.
- -Running reasoning models for eliciting ***C*** concept classes based on extracted ***F*** features.
- -Classifying ***C*** according to network concerns (application, communication and resource concern) based on the implemented and designed reasoning models and models' outcome.
- -Operating, for each set of ***C*** related to a specific network concern, the embedded behavior estimation model, discussed shortly, for investigating ***C*** similarity with already kept similar classes in the storage memory, if found, on the basis of newly extracted ***C*** features.
- -Storing/updating ***C*** classes with obtained FBS features under their main classified concern where the outcome from each implemented and designed reasoning model will specify classes registered as parent classes and sub-classes kept as child classes for the parent ones (forming DNCO).

The operation of building and updating DNCO in NetMem with a hierarchy of NMemAgents is described as follows:

- -NMemAgents in the lowest NetMem level gather raw data, via using sensory systems or management protocols like SNMP, from heterogeneous data sources, such as Internet traffic, Management Information Bases (MIB) and offline databases. Agents represent uniformly and store raw data in their storage memories to enable for agents in the next upper level data patterns learning and reasoning about semantics with higher level of abstraction. Agents continually update contents in their storage memories based on extracted semantics. Collected and updated data will be available to be retrieved according to their storage duration. Agents send other agents in the next upper level extracted semantics.
- -NMemAgents will keep and represent collected data/semantics from lower agents (and updates from peers) in their storage memories to learn semantics patterns and reason about other semantics with higher level of abstraction than underneath agents. Extracted semantics will be kept in agents' memories as associated concept classes forming DNCO. Agents continually update contents (i.e., semantics) of DNCO based on discovered differences between novel extracted semantics and already maintained ones. Agents will send next upper level agents extracted semantics.
- -Process of extracting semantics with higher levels of abstraction and forming DNCO at agents' memories is repeated until reaching the highest NetMem level where comprised NMemAgents will have DNCO comprising semantics with the highest abstraction level. Extracted semantics in the highest NMemAgents will possess the longest storage duration compared with the one of extracted semantics in lower agents. Agents in the highest level can communicate with their peers in the same level to update their maintained semantics and consequently DNCO.

3.6.3 Data/Semantics Retrieval from DNCO

The process of retrieving data/semantics from DNCO in NMemAgnet comprises the following:

- -Receiving data/semantics queries from Internet elements and/or other NMemAgents.
- -Extracting and identifying requested FBS aspects.
- -Operating the behavior estimation model to get required data/semantics and related FBS aspects via applying matching processes with already maintained data/semantics in the formed DNCO.
- -Sending required data/semantics to requesting elements or agents if matching results achieve higher degrees of similarity (e.g., semi- or completely-matching data/semantics).
- -Otherwise, directing data/semantics queries to neighboring agents at same level and upper level.

The operation of accessing built DNCO in NetMem with a hierarchy of NMemAgents for retrieving required semantics is described as follows:

- Internet elements (e.g., hosts) send NMemAgents in the lowest level asking about interesting data/semantics. Agents will search for required data/semantics through investigating stored data/semantics in formed DNCO at their storage memories. Agents will send hosts if they can find semantics matching according to hosts' requests. In case of agents' inability to meet semantics demands, agents will send unmet queries to peers in the same level and/or agents in the next upper level. Incapable agents might update DNCO in their memories with missing data/semantics according to responds they will receive from other agents.
- NMemAgents which receive queries will search on maintained semantics in DNCO in their memories. If agents cannot find required semantics, they will send neighboring agents in the same level and/or agents in the next upper level asking for those semantics.
- The last process might be repeated until reaching agents in the highest NetMem level.

3.6.4 Behavior Estimation Model

In the literature, different algorithms are presented to build behavior estimation models (e.g.,[96-98]) that can perform semantic matching and enrich concept ontology hierarchy in various fields like semantic web. NetMem adapted the behavior estimation model presented in [97] since it suits NetMem operations where that model can support semantic services via investigating concept similarity based on semantic distance. In addition, the model can consider the inheritance relations amongst concepts and their level in ontology hierarchy. SM in each NMemAgent utilizes the behavior estimation model to investigate the similarity value between newly extracted data semantics or concept classes and already stored ones in built DNCO in order to update classes and invoke actions (e.g., sending alerts when extracted semantics for unfamiliar service match behavior aspects of known attacks). Based on the value of similarity, the defined similarity threshold (T_{hd}) and the already stored concept classes in the storage memory, SM will update and/or add new concept classes to the DNCO.

For example, assuming that the model is designed to estimate and classify behavior of TCP- and UDP-based services. The usage of ARL and FMF (with defined attributes' membership degrees (MD)) aids in learning and classifying attributes related to TCP and UDP services' raw data. For instance, a TCP data profile might yield classified attributes like: "large packet size", "normal port number" and "TCP data packet type". Thereafter, learning classified latent features through using reasoning algorithms, e.g., LDA and HMM, by SM will develop a concept class. For instance, a concept class "TCP-based service" might have these features: "normal TCP packet", "communication-service port number" and "reliable communication-protocol". Then, the estimation model evaluates the similarity between the extracted concept class and already, if found, maintained classes based on the defined T_{hd} for similarity. The result of the evaluation process will lead to

update/define in the storage memory a concept class with group of features, which refers to behavior of each communication protocol-based service. We have two cases based on the estimation process's results. The first one is at having no matching concept classes in the storage memory (i.e., similarity below $\boldsymbol{T_{hd}}$). Hence, SM will define a communication concept class "TCP-based service" in the memory showing its related features as described previously. The other case is at having semi- or completely-matching classes in the memory. Then, SM will go under the top parent concept class (it might be "communication-protocol-based service") and defines/update child concept classes and features. The adopted behavior estimation model executes the following calculations:

- *Calculation of concept class weight (C_w):*

A concept class in DNCO will have $\boldsymbol{n}$ features where each comprised feature has a weight which equals $1/\boldsymbol{n}$. Each feature will have MD depending on the probabilistic and statistical analysis performed by the semantics reasoning algorithms (e.g., LDA and HMM algorithms) and features extraction algorithms on the learned attributes and the defined attribute-semantic topic association probabilities by experts. For instance, "TCP packet" feature might have two MD options which are [(normal, 0.8), (abnormal, 0.2)]. So, if we have a "normal TCP packet", the related MD value will equal 0.8.

$$C_w = \sum_f (feature\ weight \times feature\ membership) \quad (3.1)$$

The maximum concept class weight is normalized to be one.

- *Calculation of similarity index (S_i)*

$$S_i = C^D{}_w - (N((f)^D \cap (f)_{max}{}^R)/N((f)^D)) \times C^R{}_{w(max)} = S_i(C^D{}_w, C^R{}_w) \quad (3.2)$$

Where $C^D{}_w$ is the weight of discovered concept class and $C^R{}_w$ is the weight of retrieved concept class from the storage memory. $C^R{}_{w(max)}$ is the weight of the retrieved concept class which has maximum number of common features with the discovered concept class. N(f) is a function which returns number of features per concept class. $N((f)^D \cap (f)_{max}{}^R)$ should be > 0 and it returns number of common features between discovered and retrieved concept classes.

- *Calculation of similarity function (F_s)*

$$F_s = 1/(p \times |S_i| + 1) \quad (3.3)$$

This equation describes degree of concept classes matching based on the value of S_i. p is called the impact degree of concept class distance to concept similarity [97] where $0 < p \leq 1$.

Example

NetMem behavior estimation model is utilized to estimate and classify concept classes related to behavior of TCP-based services. Formerly, NetMem adopts ARL algorithms and FMF (with defined attributes' MD) to learn and classify data attributes related to TCP

services' raw data profiles. For instance, a TCP data profile might yield classified attributes like: "large packet size", "abnormal port number" and "TCP-SYN packet type". Thereafter, learning classified latent features through using reasoning algorithms, e.g., LDA or HMM, and features classification algorithms by SM will develop a concept class which might be "$\mathbf{C_{new}}$: TCP_SYN_Flood_Attack" with the following features: "large number of TCP packets", "abnormal TCP packet size", "unknown port number" and "abnormal ratio of TCP SYN packet". SM will investigate similar concept classes in existing DNCO maintained in storage memory.

SM finds in the memory a formed DNCO, shown in Figure 3.19, which focuses on concept classes related to behavior of file transfer services. Each output concept class (C) at every level of the DNCO is represented by a group of input features (F). Concept classes at upper levels are parent concept classes for classes at the lower levels. We have the following defined concept classes in the formed DNCO as illustrated in Figure 3.19:

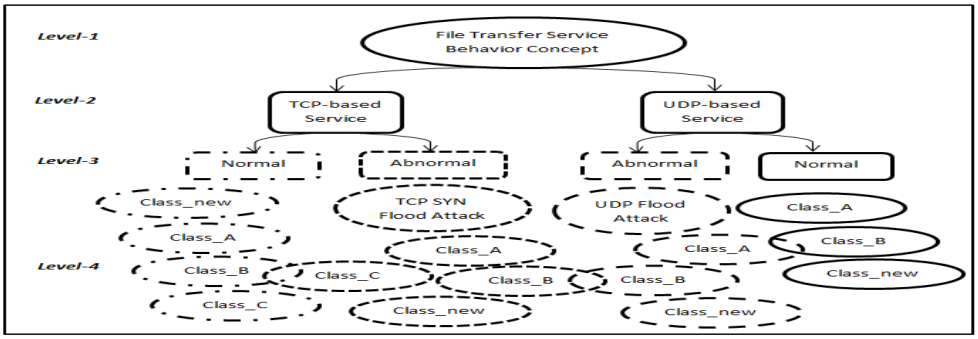


Figure 3.19 Ontology of associated concept classes

C:={ $\mathbf{C_1}$=File_Transfer_Service_Behavior, $\mathbf{C_2}$= TCP_based_Service, $\mathbf{C_3}$= UDP_based_Service, $\mathbf{C_4}$=Normal_TCP_based_Service,$\mathbf{C_5}$ = Normal_UDP_based_Service, $\mathbf{C_6}$=Attack_TCP_based_Service,$\mathbf{C_5}$=Attack_UDP_based_Service,$\mathbf{C_8}$ = TCP_SYN_Flood_Attack, $\mathbf{C_9}$= UDP_Flood_Attack, $\mathbf{C_{10}}$=Normal_TCP_service_class_A, $\mathbf{C_{11}}$=Normal_UDP_service_class_B ,……., $\mathbf{C_k}$}.

The related defined features for known concept classes might be F:={ $\mathbf{f^1}$= large number of TCP packets, $\mathbf{f^2}$= large number of UDP packets, $\mathbf{f^3}$= small number of TCP packets, $\mathbf{f^4}$= small number of UDP packets, $\mathbf{f^5}$= normal TCP packet size, $\mathbf{f^6}$= abnormal TCP packet size, $\mathbf{f^7}$= known port number, $\mathbf{f^8}$= unknown port number, $\mathbf{f^9}$= normal sequence numbers in source/destination messages, $\mathbf{f^{10}}$= abnormal sequence numbers in source/destination messages, $\mathbf{f^{11}}$=normal ratio of TCP SYN packet,$\mathbf{f^{12}}$= abnormal ratio of TCP SYN packet,……., $\mathbf{f^n}$}

The new extracted concept class ($\mathbf{C_{new}}$=TCP_SYN_Flood_Attack) by SM will be compared with defined concept class $\mathbf{C_8}$ for abnormal TCP-based service behavior in DNCO ($\mathbf{C_8}$ = TCP_SYN_Flood_Attack). This $\mathbf{C_8}$ concept class will be retrieved from DNCO for investigating similarity. According to the implemented estimation model, $\mathbf{C_8}$ will be the retrieved class C^R and it has the following features: $\mathbf{f^1}$, $\mathbf{f^6}$, $\mathbf{f^8}$, $\mathbf{f^{12}}$ with crisp Fuzzy membership functions (i.e., features have membership degree (MD) equals one

and C^R_w = one). For deciding adding new concept classes or updating already kept classes in DNCO, we define a threshold ($\boldsymbol{T_{hd}}$) for similarity where $\boldsymbol{T_{hd}}$ equals 0.9. Similarity matching processes for concept classes that yield similarity degrees below $\boldsymbol{T_{hd}}$ will result in new defined concept classes in DNCO. NetMem runs the behavior estimation model to evaluate the similarity between $\mathbf{C_{new}}$ (or C^D) and already maintained classes in DNCO. We have the following cases, based on the designed estimation model, to add $\mathbf{C_{new}}$ or update similar classes in DNCO:

- $\mathbf{C_{new}}$ has the same four features as $\mathbf{C_8}$ with similar features MD. Comparing $\mathbf{C_{new}}$ and $\mathbf{C_8}$ (or the discovered class C^D and the retrieved one C^R) yields the following calculation: $C^D_w = 4 \times (1/4) = 1$, $S_i(C^D_w, C^R_w) = 0$, $F_s = 1$. Accordingly, the new discovered concept class $\mathbf{C_{new}}$ coincides completely with already existing level-4 concept class $\mathbf{C_8}$ in DNCO. So, there is no update will be done for DNCO concept classes.
- $\mathbf{C_{new}}$ has same four features as $\mathbf{C_8}$ with dissimilar features MD. The comparison process yields the following calculation: $C^D_w = (1/4) \times (0.8) + (1/4) \times (0.9) + (1/4) \times (0.75) + (1/4) \times (0.9) = 0.8375$, $S_i(C^D_w, C^R_w) = -0.1625$, $F_s = 0.9685$. Accordingly, the new discovered concept class O_4 has a degree of similarity with already existing level-4 concept class C_8 in DNCO. So, updates might be occurred for the already defined $\mathbf{C_8}$ in DNCO.
- $\mathbf{C_{new}}$ has one common feature with $\mathbf{C_8}$ and with dissimilar features MD. The comparison process yields the following calculation: $C^D_w = (1/4) \times (0.8) + (1/4) \times (0.9) + (1/4) \times (0.75) + (1/4) \times (0.9) = 0.8375$, $S_i(C^D_w, C^R_w) = 0.8375 - 0.25 \times 1 = 0.5875$, $F_s = 0.8948$. Accordingly, the new discovered concept class $\mathbf{C_{new}}$ has a lower degree of similarity with already existing level-4 concept class C_8 in DNCO. So, a new concept class of TCP_SYN_Flood_Attack (i.e., TCP_SYN_Flood_Attack_ class_new) might be defined in DNCO since $F_s < \boldsymbol{T_{hd}}$.

3.7 Conclusion

In this chapter, we presented NetMem, a distributed network memory management system. We provided distributed multi-agent system architecture for NetMem to reason about network-semantics and develop dynamic network-concept ontology (DNCO) of correlated concept classes related to various Internet elements. We discussed the architecture of NetMem agents, or NMemAgents with composable and cooperating building block integrating: a) data virtualization and data dimensionality reduction techniques for homogenizing data originating from heterogeneous sources and reducing data dimensionality, b) cloud-like data storage techniques (big relational tables), c) associative rule learning algorithms and Fuzzy membership functions to discover and classify data attributes, d) feature extraction and classification techniques to learn and classify data features, and e) reasoning algorithms and behavior estimation models for extracting semantics and building DNCO.

Chapter 4

4 SEMANTICS MANAGEMENT IN NETMEM

In this chapter, we present discuss semantics management process and data/semantics storage/retrieval in NetMem.

4.1 Introduction

Monolithic and hybrid intelligence techniques can be used to design reasoning models for NMemAgents to manage network-semantics and build dynamic network-concept ontology (DNCO). The capability (e.g., latent features extraction ability, high prediction accuracy, etc.) of adopted reasoning model for learning rich semantics depends on the operation performance of the used intelligence techniques. In the following subsections, we discuss different semantics reasoning models, which can be implemented in each NMemAgent. Various monolithic intelligence techniques using Latent Dirichlet Allocation (LDA) [23] and Hidden Markov Models (HMM) [24] are presented for designing semantics reasoning models. Additionally, we propose hybrid intelligence technique (HIT)-based reasoning model integrating LDA and HMM for NMemAgents to efficiently extract semantics and know high-level data features. We will show characteristics and capabilities of adopted reasoning models clarifying their advantages and limitations.

4.2 Monolithic Intelligence Technique-based Reasoners

In this section, we discuss reasoning models implemented for NMemAgents using monolithic intelligence techniques.

4.2.1 Simple Statistical-Analysis-based Reasoner

Characteristics of big data, such as massive volume and complexity, impede regular data monitoring and analysis tools to anticipate data contents and structure and to interpret patterns meaningfully. Construction of statistical models [44, 99] can help in understanding patterns of big data. This would lead to a capability of extracting features and semantics. One of the problems with those models is that they are specific to certain semantic topics (i.e., those models have limitations in extracting latent features). Inefficient design (e.g., inadequate algorithmic model) for those models can lead to incorrect extracted information. There is a need for a data training phase to test accuracy of models. Some rules (e.g., Fuzzy rules) can be constructed to fit certain semantic topics. Adoption of classification techniques with rules can help statistical models to extract high level data features.

We provide a simple statistical-analysis-based model (via statistics and rules) for SM to extract semantics related to various Internet elements (e.g., behavior of TCP protocol or TCP hosts' storage memory). Figure 4.1 illustrates the proposed model. SM would learn patterns of N different data profiles represented by DVA in the storage memory to derive semantics with higher levels of abstraction. There are K targeted attributes that can be extracted and classified from stored profiles. Using data pattern learning algorithms (e.g., ARL [21]) and a classification technique (e.g., FMF [22]), SM will learn group of attributes (A_p) per each data profile in the storage memory that match required K attributes. An assumption is made that attributes per profiles are independent with equal probabilities of existence at each analyzed profile (i.e., attributes have same weights per profile). SM searches every reasoning time period (t_R) for similar data profiles (N_{Pn}) of each (n) profile of total (N) profiles. Equations (4.1) and (4.2) describe the initial attribute weight and data profile weight, respectively.

$I\_A_{i,Pn}$ is the initial attribute i weight per data profile P_n where:

$$I\_A_{i,Pn} = 1/K \qquad (4.1)$$

and $1 \leq n \leq N$, N number of profiles kept in the storage memory

$$\text{Data profile P weight } (W_{Pn}) = \sum_i (I\_A_{i,Pn} * M\_A_{i,Pn}) \qquad (4.2)$$

for each attribute i per data profile, where $1 \leq i \leq K$

```
K: targeted attributes
Ap: number of attributes per data profile
Pn: data profile which contains attributes
NPn: similar data profiles
Define min_support; define the minimum number of data profiles that have common attributes (based on number of captured data profiles)
For (n=2; n ≤ Ap; n++) {
        initialize count; counter for calculating number of data profiles' instances
        Generate candidate group CA of n attributes;
        For N data profiles in the storage memory do {
                For all attributes in each Pn do {
                        if (comprised attributes found in CA) count ++;
                }

        }
        if (count ≥ min_support)  K = CA ; learn group of attributes that found in most data profiles
}
For N data profiles in the storage memory do {
        For all attributes in each Pn do {
                calculate attributes membership using FMFs
        }
        calculate profile weight WPn
        }
learn group of NPn
For each NPn profiles do {
        calculate accuracy according to K, WPn , N, NPn and Ap
        apply defined Fuzzy rules for NPn with high accuracy
        generate semantics
}
Output: Semantics that are maintained as concept classes
```

Figure 4.1 Pseudo code for the simple statistical-based analysis model for semantics reasoning

Where $M\_A_{i,Pn}$ is the membership value of the attribute i in a data profile P_n calculated by defined FMFs. The previous simple equations calculate group of low level features that can be used to extract semantics. SM has definitions for sets of fuzzy rules to aid in extracting semantics. Those rules can be used, for example, in determining normal behavior of the storage memory in TCP hosts. With Fuzzy rules, SM adopts a vector (T) of thresholds, which are determined by experts or via SM experience and history. Based on results from profiles analysis process and thresholds' values, SM can abstract semantics. Figure 4.2 depicts a trapezoidal FMF used for calculating membership values of the bandwidth attribute. A simple SM accuracy in predicting and developing semantics is calculated using (4.3).

$$\text{SM Accuracy} = W_{Pn} \times (A_p/K) \times (N_{Pn}/N) \qquad (4.3)$$

Where $A_p \leq K$

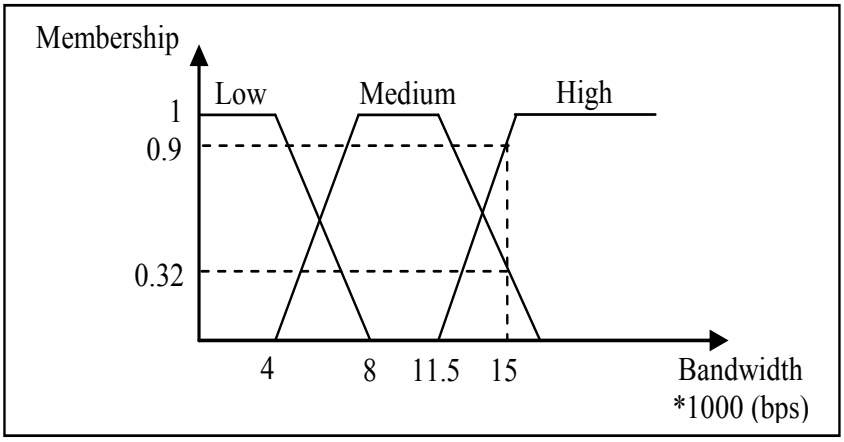


Figure 4.2 Trapezoidal fuzzy membership function for bandwidth attribute

Table 4.1 shows some statistics calculated using the above equations to learn patterns of 10 data profiles kept in the storage memory concerning a file transfer service operated by TCP protocol [100]. We assumed that three attributes, i.e., K=3, are considered in learning patterns by the ARL algorithm. Hence, SM will inspect data profiles in the storage memory and search for those attributes. Here, the problem is attribute extraction and classification (discrete target attributes) using discriminative functions, i.e., FMF. SM has definitions for Fuzzy rules which are used in, for example, determining normal behavior of the TCP communication protocol using an assigned vector (T) of thresholds, which are determined by experts or by SM via its experience and maintained history. Here is an example of a rule:

$$\textbf{\textit{IF}}\ (W_{Pn} > T_a)\&\&(N_{Pn} < T_b)\&\&((N_{Pn}/N)\times t_R < T_c)\&\&(A_p == K)$$
$$\textbf{\textit{THEN}}\ \text{normal behavior ELSE abnormal behavior}$$

T_a , T_b and T_c are thresholds defined in T vector for profile weight, number, and arrival ratio, respectively. According to the above rule, SM will extract semantics for the TCP protocol as follows:

- ***IF*** behavior = normal ***THEN*** develop semantics (S_{normal});

S_{normal} = largeNumberOfProfiles,CompleteDataProfile, NormalProfileWeight.

- ***IF*** behavior =abnormal ***THEN*** develop semantics ($S_{abnormal}$);

$S_{abnormal}$ = SmallNumberOfProfiles,InCompleteDataProfile, AbnormalProfileWeight.

largeNumberOfProfiles means that N_{Pn} exceeds the threshold T_b, CompleteDataProfile means that the data profile maintains all interesting attributes, NormalProfileWeight means that W_{Pn} is above threshold T_a. The semantics for the abnormal behavior will reveal that profiles do not satisfy the above conditions.

Figure 4.3 shows a simple semantics derived for the normal behavior of TCP protocol in storage memories of NMemAgents at high NetMem levels. For each TCP data profile that exhibits patterns similar to defined semantics, then that profile is detected as a normal one.

Table 4.1 Statistics of Different TCP Data Profiles Calculated by SM for Learning TCP Protocol Patterns

Attributes (A_p) per the service data profile			**Number of Similar Profiles (N_{Pn})**	**Total number of profiles (N)**	**Data Profile Weight (W_{Pn})**	**Profile Rank (P_r)**	**SM Accuracy**
Bandwidth (bps) membership	***Buffer size (packets) membership***	***Service duration (seconds) membership***					
15000 0.9 (high)	7000 1 (high)	1000 1 (high)	5	10	1	1	48.33%
10000 1 (medium)	3000 1 (low)	200 1 (low)	1	10	1	2	10%
14000 0.82 (high)	6000 0.4(medium)	500 0.5(medium)	1	10	0.57276	4	5.7276%
---	6500 0.55 (high)	1000 1 (high)	1	10	0.51615	6	3.441%
15000 0.9 (high)	5000 1 (medium)	---	1	10	0.666	5	4.22 %
12000 0.9(medium)	6000 0.4(medium)	1000 1 (high)	1	10	0.7659	3	7.659%

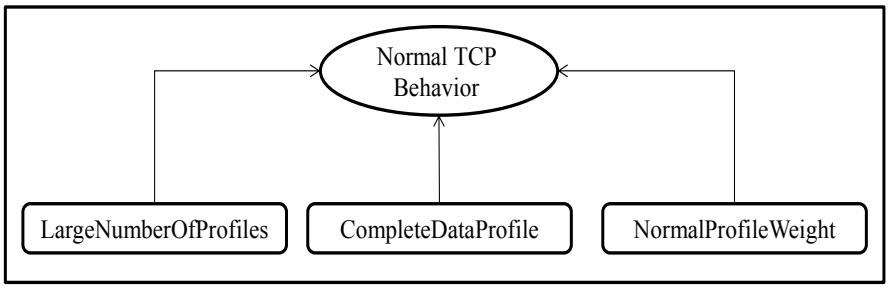


Figure 4.3 Semantics for normal TCP behavior derived by SM

4.2.2 HMM-based Reasoner

Hidden Markov models (HMM) [24, 101] are a structured architecture that is able of predicting sequences of semantic-topics based on input sequences of extracted network attributes or features. Depending on input sequences or pattern of high discriminative network-data features, HMM with forward and backward algorithms can learn semantics efficiently. HMM is widely used in learning processes and extracting information [39, 102] in different fields, such as in image and speech recognition, detection of network attacks [103], and robotics for gesture imitation [104]. HMM is a statistical Markov model for categorical sequence labeling (i.e., pattern recognition by statistical inference) based on using supervised/unsupervised learning algorithms (e.g., Baum-Welch algorithm). Sequence labeling can be treated as a set of independent classification tasks. HMM have advantages that suit operations of semantics reasoning in NetMem. HMM depends on a mathematical model with parameters (i.e., initial (π), state transition (A), and observation (B) probabilities) that can be adjusted for supporting different semantic topics in many contexts. With sets of training data, Baum-Welch's forward-backward algorithm can be applied to HMM to discover unknown HMM parameters (i.e., unsupervised learning). HMM can support multi-dimensional data (e.g., big network-data with time-based, domain-based, and service-based features). Each input state in an HMM can be specific to an output semantics domain. Considering input states as Markovian processes might affect degree of accuracy for output data. This can be mitigated, to some extent, by adjusting parameters of HMM. For example, the forward and backward transition probabilities among specific states can have the same value. This will give equal weights for get certain semantic topics if transition occurs among those states.

HMM can extract low-level data features from variable length data attributes' sequence using unsupervised learning. HMM's statistical foundations are computationally efficient and well-suited to handle new data [102]. A single HMM can be built by combining a verity of knowledge sources [104] with the consideration of their properties. This enables an efficient design of an HMM to reason about semantics related to various Internet elements. One of HMM problems is the floating-point underflow problem. This can affect extracted correct semantics. It can be overcome by taking logarithm for values of probabilities and performing summation process instead of multiplication to calculate forward probabilities. HMM have limitations in discovering high-level features with long-range syntactic dependencies. Another limitation for HMM can be found if it is required to design an HMM with large input states. This means a lot of HMM parameters to be calculated. This might affect performance (e.g., timeliness) of a semantics reasoning model implemented with HMM. HMM can be combined to form a hybrid or multi-stage model. This can aid in enhancing the performance if we can separate of feeding large scale of parameters. HMM might require large sets of data to be trained. This can be found in case of big network-data if the system can sample some sets for training. Since HMM can be considered as static models or prototypes for semantic reasoning, HMM models face challenges at operation reduced-dimension data. This will

affect accuracy of operations for obtaining right semantics. This can be mitigated via using an algorithm for latent features discovery from low and reduced dimension data. Then, those features can be supplied to HMM.

Based on the above discussion of HMM and the capability to overcome some of their limitation, we implement and test HMM within SM in NMemAgents. SM runs HMM as models for semantics reasoning and extraction. Based on extracted features and maintained data profiles by DVA in the storage memory, SM runs HMM-based reasoning models, described in Figure 4.4, after executing the training phase using sets of kept data profiles in the storage memory. The input to HMM-based models is sequence of profiles' attributes and the output is semantics (i.e., associated concepts). Extracted and classified attributes by ARL and FMF are fed as input states to SM's HMM. Based on sequence of input attributes or data features with stateless operation, calculated HMM parameters and maximum likelihood estimation, HMM can extract semantic features. These features will constitute semantics related to specific fields (e.g., behavior of Internet elements) or network concerns (application, communication, and resource).

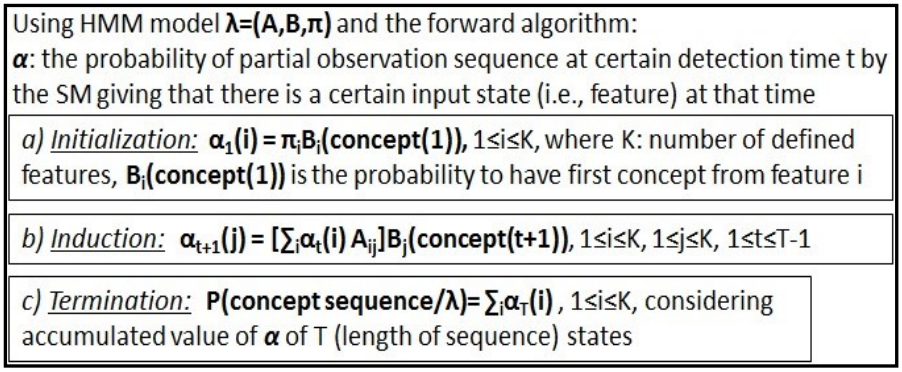


Figure 4.4 A Probabilistic model using HMM for semantics extraction

4.2.2.1 HMM Performance Measure

A HMM is defined as $\lambda = (A, B, \pi)$ with the following notation:

- $\boldsymbol{\pi}$ is the initial probability distribution for HMM states. For K discrete (feature) states {1, 2, …., K}and input sequence of T states, $\pi_i = p(state_{(t=1)}=i)$ where $1 \leq i \leq K$.
- **A** is the state transition probability matrix, which is square matrix and it shows probabilities for transition from a state (i.e., feature) to another state. $A_{ij} = p((state_{(t+1)}=j)/(state_{(t)}=i))$ where $1 \leq i \leq K$ and $1 \leq t \leq T$.
- **B** is the observable concept or semantic topic probability distribution. If there is a probability distribution over output concepts (i.e., observations) for each input feature (i.e., input state) that this distribution will show the probability to have a concept from a specific feature where there M discrete observations {1, 2, …., M}. $B_j(concept(t)) = p((concept_{(t)}=m)/(state_{(t)}=j))$ where $1 \leq m \leq M$, $1 \leq j \leq K$ and $1 \leq t \leq T$.

As mentioned in [105], the time and space complexity of a HMM are $O(K^2T)$ and $O(K^2+KT)$, respectively.

4.2.2.2 NetMem Semantics Extraction Model using HMM (via examples)

Example (1) based on collected raw network traffic data:

SM is monitoring N data profiles registered by DVA in the storage memory to learn data patterns. We assume that profiles have a length of T features, learned and their number identified by ARL, studied by statistical analysis, and classified by FMF, of group of K features that $T \leq K$. for instance, SM discovers TCP profile then it will study number of occurrence in the memory and then it will classify that number using an FMF whether it is large, or normal, or small number. We design an HMM model for SM that provides a relation between sequences of T features, extracted from learned data patterns, and generated sequences of T concepts of total M concepts based on prior transition and observation probabilities where $T \leq M$. Based on extracted features, SM extracts semantics concerning one of three networking concerns, namely, application, communication and resource.

The designed HMM λ (A, B, π) model, shown in Figure 4.5, for TCP and UDP service profiles, kept in the storage memory, has T=4 states for input (i.e., features) and output (i.e., semantics). A is defined as a matrix of input state transition probability, which shows the probability to transfer from a feature (e.g., large number of profiles) to another

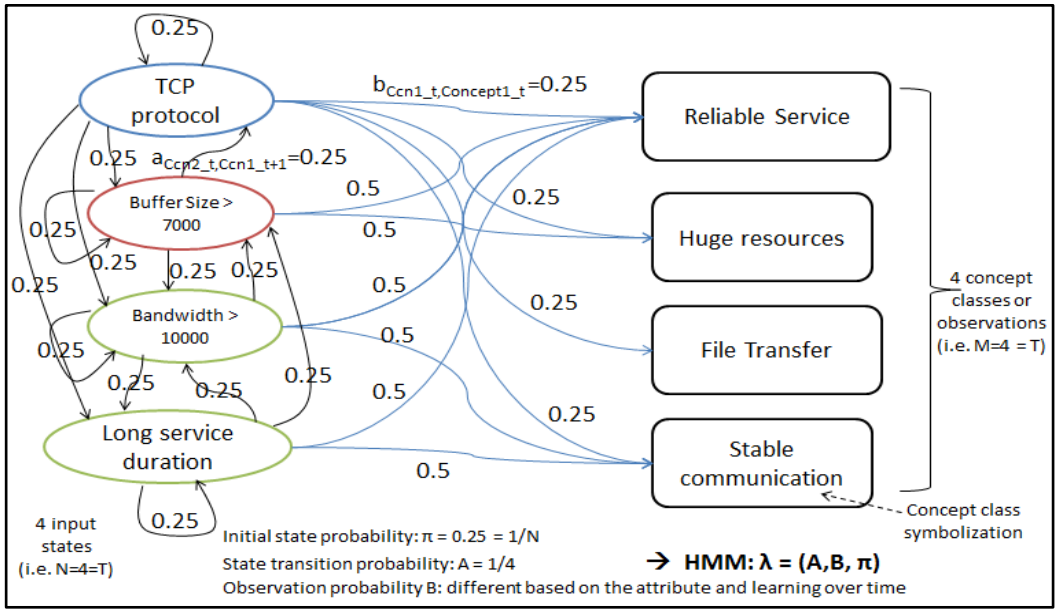


Figure 4.5 The designed HMM for reasoning about semantics related to TCP-based services

one, B is the observation probability matrix, which shows the probability to have semantics from a certain a feature, and π is the initial state probability for each feature and it equals 1/T = 0.25, i.e., T = 1/ (number of features). Four features are extracted

from TCP/UDP profiles which are: *protocol type, buffer_size, bandwidth, long_service_duration.* Figure 4.6 illustrates a trained HMM using the Baum-Welch algorithm [51] for the initial designed HMM in Figure 4.5 and via adopting 50000 sequences of TCP-based service attributes.

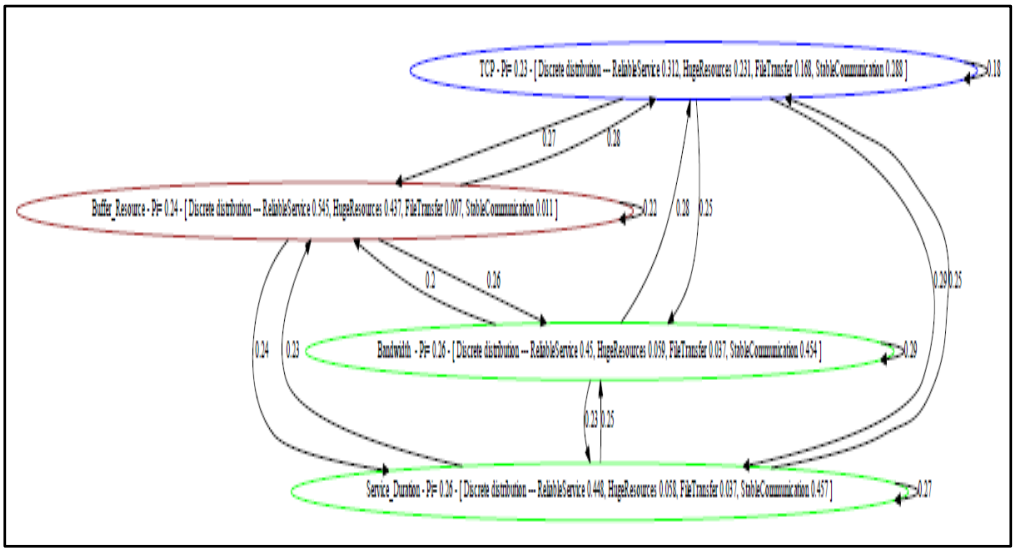


Figure 4.6 Trained HMM by the Baum-Welch algorithm

We defined decision trees [93] and FMFs that classify extracted features to show, for example, the degree of bandwidth value and range of used buffer size. Obtained features are input sequence to our HMM. Based on the trained HMM in Figure 4.6, Figure 4.7 shows an example for input sequence of state (i.e., data attributes or features) to the designed HMM. Based on training sequences and HMM parameters, we have at output a

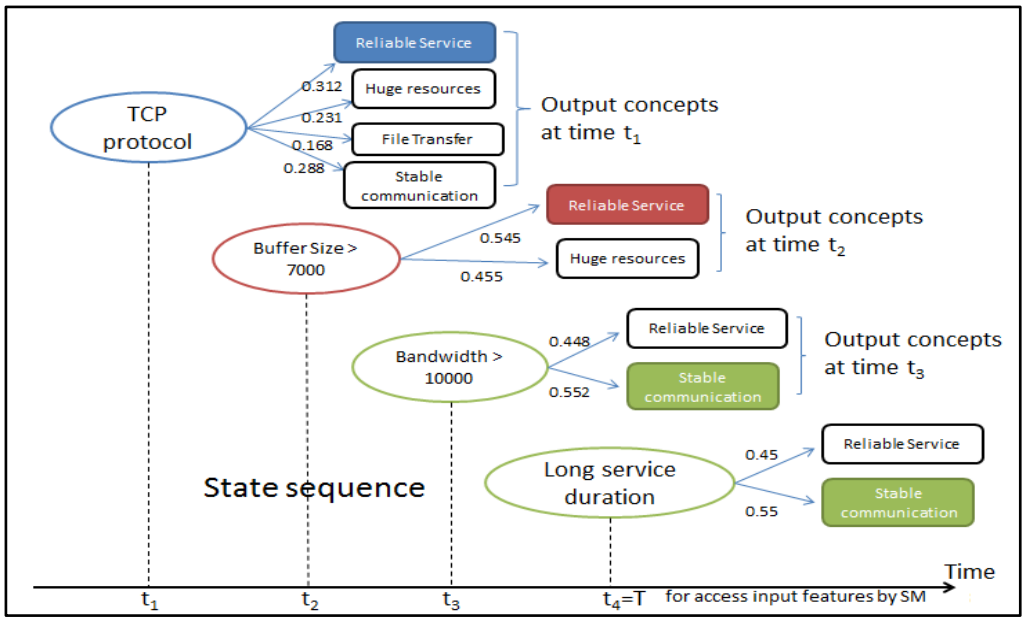


Figure 4.7 The state sequence flow to the designed HMM

sequence of four defined semantics which are *reliable service, huge resources, file transfer, stable communication*. Using the Viterbi algorithm [105] for maximum likelihood estimation, SM extracts semantics for TCP and UDP profiles at different levels of granularity defining; application concern (*file transfer* and *reliable service*) and communication concern (*stable communication*). Experts might define new concepts which will depend on input features of TCP-based services. This can enhance the granularity of estimated concepts where more association can be built that many concept classes will be derives for TCP-based services.

Example (2) based on collected raw data from management information bases (MIB) in TCP clients:

NetMem adopts HMM-based reasoning model to learn behavior semantics or concepts of storage memory in TCP hosts based on collected raw data from hosts' MIBs and formed profiles in the storage memory. Based on extracted and classified attributes by ARL and FMF, respectively, HMM has five input states for estimating the behavior of those memories. Those states are "*large memory size*", "*long memory system uptime*", "*abnormal allocation failures*", "*normal running system processes*", "*small stored session information size*". The most likelihood estimated output concept classes based on any sequence of the previously mentioned attributes are "*host resources behavior*" and "*abnormal storage memory behavior*". To have that feature (i.e., independence of input sequence form) in HMM, the state transition and observation probabilities should be designed carefully. For instance, group of input states might have equal transition probabilities to transfer from one of them to others. Also, the observation probability of a related semantic topic will be with a high value with respect to any of those attributes.

The HMM parameters are trained and assigned using the unsupervised Baum-welch learning algorithm [51]. That algorithm depends on an initial developed HMM for finding the maximum likelihood HMM parameters through iteratively training the parameters of the initial model relied on the observed output sequence. The ability of input to HMM sequence of data attributes related to diverse network concerns enables getting output sequence with associated semantic topics or concept classes at different levels of abstraction. As mentioned in the above example, an input sequence with equal initial state probability (i.e., π =1/5) to HMM might begin with one of the above aforementioned five classified attributes. We assume the designed HMM with equal state transition probabilities (i.e., A_{ij} = 1/4 for i≠j and A_{ij} = 0 for i=j where A_{ij} is the transition probability form state *i* to state *j*). To get the output concept classes, the observation probability B matrix, which relates each input state with an output, might equal, for instance,((**0.5**,0.1,0.4),(**0.55**,0,0.45),(0.1,**0.8**,0.1),(**0.7**,0,0.3) ,(**0.6**, 0.35,0.05)). B matrix consists of ***r*** rows and ***c*** columns where ***r*** equals the number of input attribute and ***c*** equals the number of output concept classes. The three available output concept classes representing the three B columns are "*host resources behavior*", "*abnormal storage*

memory behavior" and "*huge storage resources*". Here, we remark according to the previous example that input classified attributes have different observation probabilities with the possible output concept classes.

4.2.3 LDA-based Reasoner

LDA [23] is a generative probabilistic dynamic model that can be used for data semantics reasoning based on learned data attributes. LDA has the capability to discover high-level features of data profiles where it has a well-defined inference methodology to associate a data profile with group of attributes with several semantic topics. LDA is capable of extracting data semantics based on prior probabilities for data profile-semantic topic and attribute-semantic topic associations.

The operation of LDA to discover feature or semantic topics in M analyzed profiles in the storage memory is executed every defined reasoning window or through other criteria as triggering signal sent from DVA to SM. LDA samples a hidden semantic topic z for each m data profile through calculating sampled posterior probability vector θ of topic-data profile association which depends on prior association weight α, number of the m^{th} profile's attributes related to a certain topic z, and N total number of attributes in the m profile. Also, LDA calculates sampled posterior probability φ of attribute-topic association based on prior attribute-topic association weight β, number of attribute instances assigned to topic z, and total number of attributes in all M profiles assigned to topic z. Figure 4.8 shows LDA algorithm for semantics reasoning processes in NMemAgents.

Through certain number of iterations, the posterior probability to have a specific semantic topic assigned to data profiles and attributes is enforced. This is because that semantic topics with high association probabilities will have great chances to be assigned

Initialization

- Define Z semantic topics, input M profiles each has N attributes of total V attributes, α a real value Z-dimension vector (prior weight of topic z in a profile), β a real value V-dimension vector (prior weight of attribute v in a topic)
- Assign randomly a topic for each attribute in a data profile and build matrix z[M][N]

Definitions

nw[i][j]: # of instance of attribute i assigned to topic j, nd[i][j] number of attributes in document i assigned to topic j, nwsum[j] total number of attributes assigned to topic j, ndsum[i] total number of attributes in document i

Generative Process

```
For iter = 1:Iterations
    For m = 1: M (number of profiles)
        For n=1:N (number of attributes per each profile m)
            remove current state topic z from the count variables for each attribute n in profile m
            for z= 1 : Z (number of semantic topics)
                p(topic z) = (nw[profiles[m][n]][z] + β)/(nwsum[z] +
                             V * β) * (nd[m][z] + α)/(ndsum[m] + Z * α)
            end
            sample topic z from p
            add newly estimated topic z to count variables
        end
    end
end
```

Figure 4.8 LDA algorithm for semantics reasoning implemented in SM of NMemAgents

to data profiles and related attributes after finishing all iterations. For instance, assume a data profile which has prior association probabilities with four semantic topics (p(1)=0.1, p(2)=0.2,p(3)=0.3,p(4)=0.4). After using Gibbs sampling and choosing a random number (e.g., $\boldsymbol{u}$) from the total summation of all probabilities (i.e., p_tot=1), $\boldsymbol{u}$ will be less than 1. So, the assigned topic to that data profile should be for a topic that its prior association probability greater than $\boldsymbol{u}$.

LDA-based reasoning models [23] provide semantic-topics models that enable discovery of hidden topic-based patterns through supervised learning process. LDA gives a systematic and well-defined way for inferring latent semantic topics in large scale of multi-dimensional data sets. LDA assign random values for prior probabilities semantic topics and related attributes based on data sampling (e.g., Gibbs sampling [106, 107]), multivariate distribution (e.g., dirichlet distribution) and training data sets. LDA models have the capability of extracting latent features and semantics based on prior probabilities for data attribute- data profile and data attribute-semantic topic associations. LDA assumes that attributes of data are related to random chosen semantic topics. The selection of semantic topics is based on random values of multivariate probability distribution parameter. LDA executes semantic topic sampling for each data attribute in every data profile for all profiles, and the updating processes for profile-semantic topic and attribute-topic associations, which are iterated many times. LDA models have the ability to estimate semantics based on small scope of extracted features with long-range semantic dependencies. LDA supports semantic extraction from big data with high dimension. Semantic-topics selection parameters and prior probabilities can be directed to relate to specific attributes to group of topics (i.e., having specific distribution of semantic-topics over group of attributes). This enables LDA models to work with low-dimensional data, such as reduced-dimension data obtained through an LSH algorithm.

One of the main advantages for LDA models that these models are extendible[38] to support more associations amongst semantic topics and data attributes. LDA models can be used to enhance functionalities of other semantic reasoning models (e.g., HMMs) to strengthen their capability of extracting latent features and semantics. Some limitations of LDA models can be found due to their randomization process for assigning parameters' values of attribute-topic associations. This can result in inaccurate predictions and semantic topics. LDA models, at large number of training data sets, can face the overfitting problem. LDA has some limitations [108], such as the bag of words or attributes assumption where there is a possibility of semantic topic sampling process to allow attributes, related to same semantic topics, to be assigned to other semantic topics. Furthermore, the LDA's probabilistic inference process for topics is computational complexity where that process takes polynomial time in case of inferring document's topics using a small number of semantic topics [109].

According to the above discussion and characteristics of LDA models, SM can utilize LDA as generative probabilistic dynamic models for semantics reasoning. LDA models

will aid in possessing NetMem the capability of efficient data recollection based on low-dimensional data kept in the storage memory. SM assigns analyzed data profiles in the memory to topics (i.e., networking concerns as defined in [5]) based on prior assigned weights for extracted and classified features with topics and also data profiles with topics. LDA calculates the most probable topic for each profile through posterior probabilities for profile-topic associations. Extracted semantic topics will be stored as associated concept classes by DVA. NetMem operation will utilize LSH to mitigate LDA's limitation of bag of words assumption where SM will manage low-dimensional data profiles. Consequently, LDA will be better able to estimate semantic topics based on small scope of extracted attributes with long-range semantics dependencies. For example, LDA might detect a TCP-based service profile which refers to a TCP SYN flood attack (i.e., a semantic topic). The detection process depends on dimensional-reduced profiles formed by LSH and with three classified attributes, which are "SYN packet type", "small packet size", and "highly frequent profile". Those attributes have high association probability with that attack. Hence, data models using LSH and semantics reasoning models using LDA will aid in endowing NetMem or NMemAgents with the capability of efficient data recollection based on low-dimensional data stored in their storage memories.

4.2.3.1 LDA Performance Measure

Equations (4.4) and (4.5) is used to calculate the perplexity where that equation was derived based on the one mentioned at [23]. Calculating perplexity helps in evaluating performance of the LDA models in detecting and categorizing attributes in data profiles based on learned parameters (e.g., prior profile-topic association weight or probabilities). A data profile is represented by a network-data storage model where it comprises data attributes related to traffic among Internet elements. So, based on *M* data profiles and their *N* related attributes and number of defined semantic topics *K*, we can get the value of perplexity.

$$Perplexity = \exp^{-((\sum_{m=1}^{M} \log(p(attribute_m))/(\sum_{m=1}^{M} N_m))} \quad (4.4)$$

Where $p(attribute_m) = \prod_{n=1}^{N_m}(\sum_{j=1}^{K} p(attribute_n / topic_j)p(topic_j/profile_m))$ (4.5)

and N_m is the number of attributes per each data profile m.

In [109], the authors investigated the inference problem, or the maximum a posteriori (MAP) problem, at LDA. That problem relates to the most likely assignment process of semantic topics to profile's attributes. Due to that problem, it was proved in [109] that the time complexity of LDA is: a) "*polynomial*" $O((N_mK)^k(N_m+k)^3)$ in case of having small k topics appear in document (or data profile) where k<<K; and b) non-deterministic polynomial (NP)-hard in case of LDA general settings with arbitrary topics per data profiles where each profile might have K topics in its MAP assignment. Henderson *et al.* in [110], investigated the LDA inference using standard Gibbs sampling approach. Based on that approach, the runtime and space complexity of LDA with Gibbs sampling were $O(MKN_m)$ and $O(M(K+N_m))$, respectively.

4.2.3.2 NetMem Semantics Extraction Model using LDA (via examples)

Example (1) based on collected raw network traffic data:

Assume that there are 10 unique profiles stored in the storage memory. Each profile comprises seven attributes. LDA randomly allocates a semantic topic for each attribute based on initially defined weights of topic-attribute associations. Accordingly, the weight of the assigned topic with respect to related attributes is increased. For certain number of iterations, the process repeats and LDA will provide posterior weights of topic-attributes association and accordingly profile-topic association. The profile will be assigned to a specific topic (e.g., normal TCP-based service) based on assigned weights of that profile's attributes regarding the assigned topic. In other words, the profile will have large number of attributes (at least more than 4 attributes) with high weight regarding its assigned semantic topic.

Example (2) based on collected raw data from management information bases (MIB) in TCP clients:

NetMem learns normal behavior of storage memories at TCP clients. Raw data concerning host resources in MIBS are collected and maintained in the storage memory. Through a certain number of iterations and using Gibbs sampling [106, 107], LDA draws hidden topics for each *m* data profile in the memory and then draws feature topics for comprised attributes in each analyzed data profile. For example, three feature or semantic topics are defined in LDA: ("*host resources behavior*", "*abnormal storage memory behavior*" and "*huge storage resources*"). Twenty data profiles (i.e., M=20) in the storage memory has the same three extracted and classified attributes which are "*large memory size*", "*long memory system uptime*", "*abnormal allocation failures*", "*normal running system processes*", "*small stored session information size*". Each attribute and profile has a prior topic association weight vector. Based on the overall prior weight vectors *(α* and *β)* and number of semantic topics, a sampled topic association probability vector p_{assoc} of length equals the number of available semantic topics is calculated like p_{assoc}=p(semantic_topic_1)=0.6, p(s_2)=0.3, p(s_3)=0.1. In each LDA iteration, the current assigned topics for a data profile and comprised attributes are removed. Then, a random number $\boldsymbol{u}$ is sampled based on p_{assoc} and the summation of its contents. The higher *p* topic association value will be chosen and the related topic is assigned. For example, if $\boldsymbol{u}$ equals 0.4, number of attributes and related profiles assigned to the first semantic topic (i.e., the new topic) increases. Thereafter, updates will be happened to posterior association weights θ and φ according to changes in number of attributes and profiles that relate to first semantic topic. Hence, the posterior association weight of the first topic with data profiles and comprised topic-related attributes increases. In the example (stated in the HMM section) of semantic reasoning for the behavior of storage memory in TCP hosts, LDA assigns the classified attributes in analyzed data profiles to the following feature topics: "*host resources behavior*", "*abnormal storage memory*

behavior" and "*huge storage resources*". For instance, LDA assigns a data profile to a specific semantic topic "*abnormal storage memory behavior*" based on assigned weights of that profile's attributes regarding the assigned topic. More than one semantic topic can be assigned to data profiles based on the executed topic modeling process (which depends on parameters of the used multinomial distribution) and profiles' attributes.

4.3 Hybrid Intelligence Technique (HIT)-based Reasoner

4.3.1 HIT Overview

We propose and implement a HIT for NMemAgents to build efficient reasoning model to reason about network-semantics based on learning patterns of full or reduced-dimensional data. The HIT integrates LDA [23] and HMM [24]. Our HIT is designed to overcome limitations of the semantics reasoning operation with only adopting HMM, which was introduced in our preliminary NetMem design [111]. The hybridization of HMM and LDA enables latent features extraction with semantics dependencies not just based on learning features with syntax dependencies. On the one hand, LDA has the capability to discover high-level features of data profiles with full or reduced dimensionality. LDA possesses extendible operation [38] where it can support more associations amongst semantic topics and extracted high-level data features. LDA has a well-defined inference methodology to associate a data profile with groups of attributes with several semantic topics. However, adopting LDA alone in semantics reasoning might produce some shortcomings due to LDA's limitations, such as the bag of words assumption [108] that might result in semantic topics misclassification where there is a possibility of semantic topic sampling process to allow attributes, related to same semantic topics, to be assigned to other semantic topics. Furthermore, the LDA's probabilistic inference process for topics is computationally complex where that process has NP-hard complexity in case of inferring document's topics using a large number of semantic topics [109]. On the other hand, HMM can efficiently extract data semantics from variable-length input sequence of data features, extracted by LDA, using unsupervised learning algorithms [112]. HMM's statistical foundations are computationally efficient and well-suited to handle new data [102]. A single HMM can be built by combining a verity of knowledge sources [104] with the consideration of their properties. This enables an efficient design of an HMM to reason about semantics related to various Internet elements. However, utilizing HMM singularly for semantic reasoning might result in some deficiencies due to HMM incapability: a) to discover high-level features with long-term semantics dependencies due to the assumption about data using the Markovian process and the usage of maximum likelihood estimator [113]; and b) to work efficiently with reduced-dimension data since HMM needs large amounts of data for training and adjusting HMM's unstructured parameters.

We propose the HIT to overcome limitations of the operation with only HMM or LDA, which was used in our preliminary NetMem design [114]. Integrating LDA with

HMM enables latent features extraction with semantics dependencies not just based on learning features with syntax dependencies. Our proposed HIT can enhance the operation of LDA by assuming Markovianity of attribute sequences [23] which can mitigate the LDA limitation caused by the bag of words assumption [108] . Also, the designed HIT will consider advantages of comprised intelligence techniques; HMM and LDA. We are motivated in our HIT design by the capability of LDA to discover latent high-level features from multi-dimensional network-data patterns with long-range semantics dependencies. Those features will be grouped as sequences that enable semantics reasoning operation via HMM with higher accuracy. LDA models are extendible [38] in that they can support more associations amongst semantic topics and extracted data features. Hence, HMM will be able to efficiently reason about data semantics related to input sequences of high-level data features. HMM [24] are structured architectures that are able to predicting sequences of semantic-topics based on input sequences of extracted network attributes or features. Depending on input sequences or pattern of high discriminative network-data features, HMM with forward and backward algorithms can learn semantics efficiently. The output from feature extraction process executed by LDA (i.e., input to HMM) is random based on analyzed data. This agrees with the characteristics of HMM to predict output based on hidden data sequences. Figure 4.9 shows SM of NMemAgents with the hybrid semantics reasoning model.

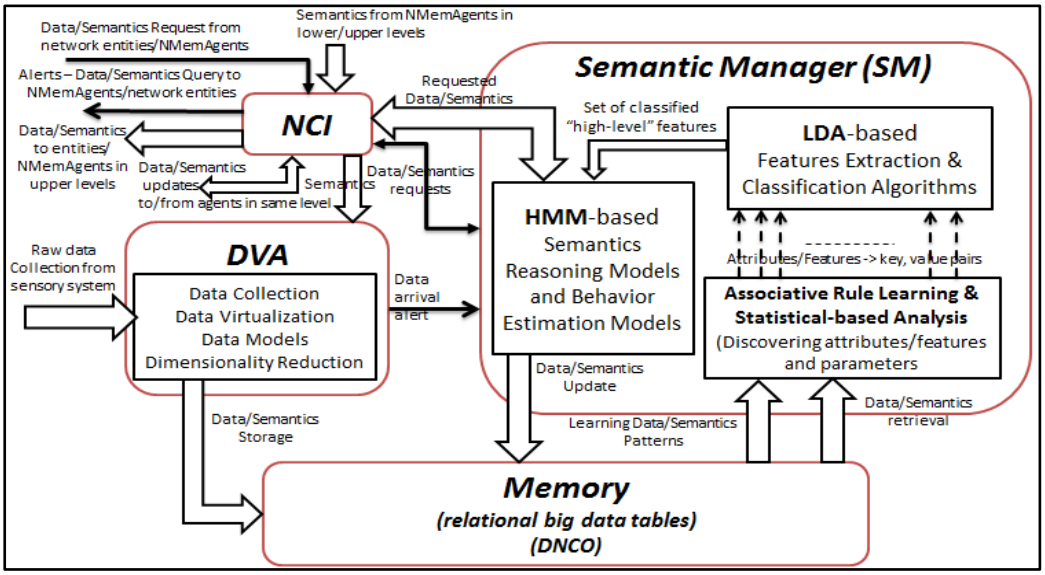


Figure 4.9 NMemAgent with the hybrid model for semantics reasoning

4.3.2 Semantics Reasoning Process using the HIT

SM task depends on learning patterns of multi-dimensional dynamic data in the storage memory. Based on learned patterns, SM will reason about semantics and maintain them as associated and classified concepts. Figure 4.10 shows the pseudo code of the HIT-based semantics reasoning process which is executed by SM. SM learns group of

attributes in data profiles kept in the storage memory through utilizing associative rule learning [21] and simple statistical-analysis-based models using FMF [22]. Construction of statistical models [44] can help in understanding patterns and analyzing low-level features of big data. SM adopts LDA for extracting and classifying high-level data features with long-range semantics dependencies. SM integrates the capabilities of LDA with HMM for semantics reasoning. Integration of LDA and HMM forms a hybrid model for semantics reasoning in NetMem. LDA are extendible [38] that they can support more associations amongst semantic topics and extracted data features. Classified features by LDA are input to HMM as sequences. While SM is learning patterns, SM might access already maintained concepts to test if there are matching concepts or not. This enables NetMem to predict future events and learn behavior and requirements of novel Internet elements. Data models utilized by both DVA and SM show the FBS aspects of maintained data and concepts through using the FBS engineering design framework [19, 20].

Algorithm: Integrated LDA-HMM model for semantics reasoning

Input: operation time t and reasoning period t_R, LDA-parameters: Z hidden topics, K feature topics and hyperparameters α & β, Gibbs sampling iterations J, HMM λ-model parameters (A,B,π) for N states & T semantic topics

Repeat every t_R
  Attribute corpus a = {a^1, a^2, ……, a^L} ← getAttributes()
  Data/Concept Class Profiles corpus P = {P^1, P^2, ……, P^M} ← captureDataConceptClassProfiles()
  for j = 1:J **do**
    for m = 1:M **do**
      sample topic z of Z topics for the m^{th} data profile based on Gibbs sampling & prior probabilities α & β
    end for
  end for
  calculate θ^m = Dir(α) for Z feature topics in each data profile & **calculate** φ^z = Dir(β) for attributes in each m profile
  for m = 1:M **do**
    draw z^m ~ multinomial(θ^m)
    for n = 1:N (where N of attributes per profiles & N ≤ L) **do**
      sample feature topic f_n^m from K topics for each n attribute in the m^{th} data profile, draw f_n^m ~ multinomial(φ^{z^m})
    end for
  end for
  initialize HMM models and run Baum-Welch algorithm for learning models' parameters
  for m = 1:M
    for i = 1:T
      calculate observation probability $B_{f_n^m}$(semantic-topic$_i$) for each n input state (i.e., classified feature f_n)
    end for
    get maximum likelihood semantic-topics sequence of length T with high probability for each implemented HMM
  end for
until operation time

Output: set of semantic topics that are related to various concerns and represented as assocaited concept classes

Figure 4.10 Pseudo code of the HIT-based semantic reasoning model

Figure 4.11 describes the semantics extraction process implemented by the LDA-HMM-based reasoning model in SM. Also, the section shows a behavior estimation model used by NMemAgents to classify behavior concepts and to build ontology of concepts in their storage memories. Network-data characteristics include massive volume, high- and multi-dimensionality, dynamicity, complexity (variety in representation models and languages). In [52], authors proposed a generative model based on Latent Dirichlet Allocation (LDA) [23] and HMM for learning words with short-range syntax and long-range semantics dependencies. Consequently, this aids in

forming richer ontology with more associated semantic topics and classes. There is similarity between characteristics (e.g., huge volume, high dimensionality, and complexity) of datasets in networks and language modeling. So, we provide the hybrid LDA-HMM-based reasoning model integrating LDA and HMM for combining the advantages of both algorithms in efficiently: a) learning patterns of big data with reduced-/high- and multi-dimensionality; and b) building dynamic network-concept ontology (DNCO) showing different and correlated concept classes [94].

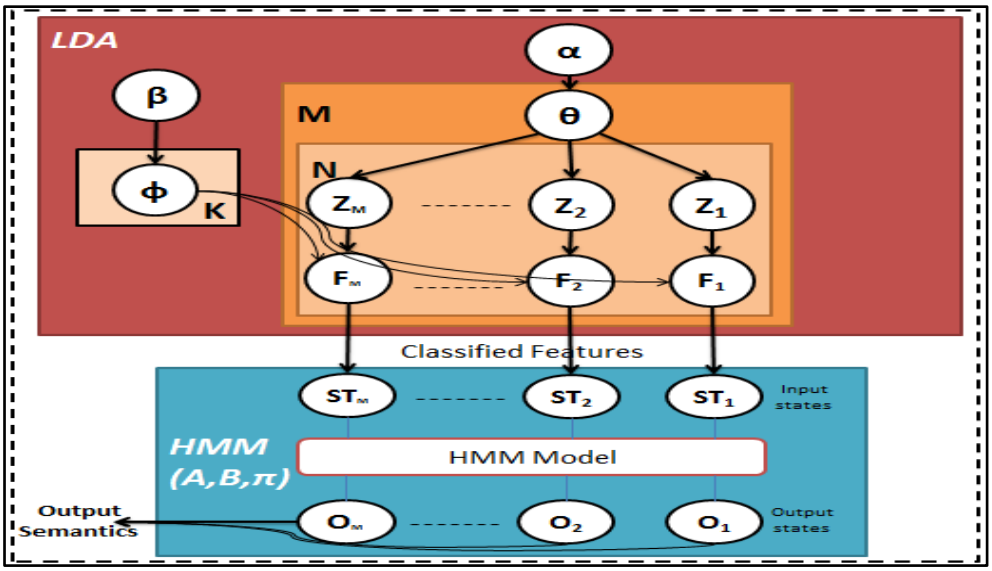


Figure 4.11 The LDA-HMM-based Reasoning Model

We provide a semantic reasoning model for NMemAgents using the proposed HIT in order to have highly abstracted and associated semantics at different levels of granularity and relate to various network concerns. At using HIT-based reasoning model, LDA is able to extract latent features with long-range semantics dependencies based on adopting inference models. Those models define correlations of semantic topics among data attributes and related data profiles. Relied on determined inference models, LDA can extract hidden semantic topics related to each data profile's attribute and also assign an overall topic to every data profile. HMM, as individual reasoning techniques, can be used to extract low-level features with short-range semantics dependencies [112]. Based on extracted semantic topics or features by LDA, HMM output depends on the sequence of input data features related to diverse network concerns. HMM can be designed and trained to get output based on sequences of data features, independent of features' order. In other words, the outcome from a HMM-based reasoning process can focus only on the existence of certain data features regardless of features' order.

We are motivated in our HIT design by the capability of LDA to discover latent high-level features from multi-dimensional network-data patterns with long-range semantics dependencies. Using correlations of semantic topics among data attributes, LDA can extract hidden semantic topics related to each data attribute and also assign an overall

topic of a group of attributes. Those classified semantic topics of features will be sequenced to enable semantic reasoning via HMM with higher accuracy. HMM are structured architectures that are able to predict sequences of semantic topics (related to different network concerns) based on input sequences of extracted network features by LDA. Depending on input sequences or pattern of highly-discriminative network-data features, HMM with forward and backward algorithms can learn semantics efficiently showing their FBS aspects.

Some related work (e.g., [61, 67, 69, 77]) adopted monolithic and hybrid techniques for enhancing networking operations, such as intrusion detection and efficient routing. However, those works were application-specific and they did not provide a way for building ontology of associated concept classes related to various Internet elements (e.g., applications and services). Also, unlike [70, 75], NetMem provides scalable memory storage for raw data and extracted semantics. Highly dynamic raw data and semantics with lower levels of abstraction are kept in storage memories of low level NMemAgents enabling learning patterns of data related to different network concerns. Little varying semantics with higher levels of abstraction are maintained in storage memories of high level agents offering long-term access for associated concept classes.

4.3.3 Operation of HIT-based Reasoner

In our HIT-based reasoning model, LDA [23] is able to extract latent features with long-range semantics dependencies based on adopting inference models. Those models define correlations of semantic topics among data attributes and related data profiles. Relying upon determined inference models, LDA can extract hidden semantic topics related to each data profile's attribute and also assign an overall topic to every data profile. Based on extracted semantic topics or features by LDA, HMM [24] output depends on the sequence of input data features related to diverse network concerns. HMM can be designed and trained to get output based on sequences of data features, independent of features' order. Table 4.2 shows the parameters used within the HIT.

Table 4.2 The LDA-HMM-based model's parameters

Symbol	Description
K	Number of feature topics
Z	Identity of hidden semantic topics
V	Number of attributes in all data profiles (or profiles)
N	Number of attributes per each data profiles, $N \leq V$
M	Number of data profiles
F	Identity of feature topics of all attributes
α	Prior feature topics/profile weight for feature topic-profile association
β	Prior attribute/topic weight for attribute-feature topic association
θ	Sampled posterior feature topics/profile weight vector of length Z for feature topic-profile association
φ	Sampled posterior attribute/profile weight vector of length V for profile attribute-feature topic association
π	Initial state probability vector of length N

A	State transition probability matrix of size N×N
B	Observation (or output semantics) probability matrix of size N×N
ST_i	Input HMM state i (or feature topic) , $i \leq N$
O_i	Output semantics based on input state i

The semantic reasoning process in each NMemAgent is executed every defined reasoning window size or through other criteria as triggering signal sent from DVA to SM. The HIT operation depends on learning patterns from profiles and comprised attributes in the storage memory. Groups of attributes are discovered using the applied ARL algorithm and a defined set of FMF in SM. For instance and through a defined reasoning window, SM aggregates information, initially, via learning patterns of TCP data profiles through recognizing and classifying attributes of each profile. Analyzing TCP data profiles by ARL and FMF might give for LDA the following classified attributes: 10000 profile's instances, large_TCP_packet_size, TCP-SYN_packet_type, file_transfer_ service_type.

4.3.3.1 LDA operation in HIT

LDA [23] gives a systematic and well-defined way for inferring latent semantic topics in large scale of multi-dimensional data sets. LDA assigns random values for prior probabilities of semantic topics and related attributes based on data sampling (e.g., Gibbs sampling), multivariate distribution (e.g., dirichlet distribution) and training data sets. LDA has the capability of extracting latent features and semantics based on prior probabilities, using dirichlet distribution, for data-attribute data profile and data-attribute semantic topic associations. LDA assumes that attributes of data are related to randomly chosen semantic topics. The selection of semantic topics is based on random values of multivariate probability distribution parameters. LDA executes semantic topic sampling for each data attribute in every data profile for all profiles, and the updating processes for profile-semantic topic and attribute-topic associations, which are iterated many times. LDA looks at each data attribute and generates its related latent feature within each data profile (i.e., LDA makes semantic topic modeling for each data profile based on its comprised attributes). LDA randomly assigns a semantic topic for each attribute based on initially defined weights of topic-attribute associations. Accordingly, the weight of the assigned topic with respect to related attributes is increased. For certain number of iterations, the process repeats and LDA will provide posterior weights of topic-attributes association and accordingly profile-topic association.

Through a certain number of iterations and using Gibbs sampling [107], LDA extracts and classifies high-level features associated with each analyzed m data profile of total M profiles in the storage memory. LDA samples a hidden semantic topic z for each m data profile through calculating sampled posterior probability vector θ of topic-data profile association which depends on prior association weight α, number of the m^{th} profile's attributes related to a certain topic z, and total number of attributes in the m profile. Also, LDA calculates sampled posterior probability φ of attribute-topic association based on

prior attribute-topic association weight β, number of attribute instances assigned to topic z, and total number of attributes in all M profiles assigned to topic z. For example, three feature or semantic topics (i.e., K=3) are defined in LDA: ("normal TCP packet", "normal comm-flow", "TCP comm-protocol"). Ten data profiles (M=10) in the storage memory have the same three attributes (i.e., N=V=3). Each attribute and profile has a prior topic association weight vector. Based on the overall prior weight vectors *(α* and *β)* and number of semantic topics, a sampled topic association probability vector p_{assoc} of length equals the number of available semantic topics is calculated like p_{assoc}=p(semantic_topic_1)=0.75, p(s_2)=0.2, p(s_3)=0.05. In each LDA iteration, the current assigned topics for a data profile and comprised attributes are removed. Then, a random number $\boldsymbol{u}$ is sampled based on p_{assoc} and the summation of its contents. The higher p topic association value will be chosen and the related topic is assigned. For example, if $\boldsymbol{u}$ equals 0.6, number of attributes and related profiles assigned to the first semantic topic (i.e., the new topic) increases since p(s_1) which equals 0.75 is greater than 0.6. Thereafter, updates will be happened to posterior association weights θ and φ according to changes in number of attributes and profiles that relate to first semantic topic. Hence, the posterior association weight of the first topic with data profiles and comprised topic-related attributes increases.

4.3.3.2 HMM operation in HIT

HMM [24] comprises categorical sequence labeling supervised/unsupervised algorithms for estimating observations based on sequence of hidden input words (i.e, data attributes or features). Extracted and classified features, output from LDA, form a sequence and convey to parameters of HMM to generate semantics. The estimation process for HMM observations relies on continuous input sequence with different Gaussian distributions. Then, HMM performs distribution mixture for obtaining the most likelihood output sequence. HMM looks at the group and the sequence of data attributes or features. The order of states in an input sequence might change the output observations. In other words, the existence of the same data features, however, with different order might result in different outputs adopting the same HMM. However, the HMM can be designed to have outputs based on having specific group of input states (or features) without considering their sequence order.

The HMM parameters (A, B, π), discussed shortly, are trained and assigned using the unsupervised Baum-welch learning algorithm [51]. That algorithm depends on an initial developed HMM for finding the maximum likelihood HMM parameters through iteratively training the parameters of the initial model relied on the observed output sequence. The ability of input to HMM sequence of data features related to diverse network concerns enables getting output sequence with associated semantic topics or concept classes at different levels of abstraction. For example, an input sequence to HMM might be (*"normal TCP packet size", "normal comm-flow", "TCP comm-protocol"*) with equal initial state probability π (i.e., π =1/3) and state transition

probabilities A (i.e., A_{ij} = 1/2 for i≠j and A_{ij} = 0 for i=j where A_{ij} is the transition probability form state *i* to state *j*). The first feature can be classified as an application concern and the other two features as communication concerns. Accordingly, the expected HMM output observation based on the previous sequence with any feature order might be "normal TCP-based service". To get the previous output, the observation probability B matrix, which relates each input state with an output, regarding that concept class (i.e., output) will be high. For instance, B matrix might consist of three rows ***r*** and three columns ***c***; and it might equal ((0.3,**0.5**,0.2),(0.2,**0.8**,0.0), (0.4,**0.45**,0.05)) where the number of ***r*** equals the number of input features and the number of ***c*** equals the number of output concept classes. According to the previous example, all input features have high observation probability with the "normal TCP-based service" concept.

4.4 HIT-based versus Monolithic Reasoners

We highlight in this subsection, via working example, the differences between NetMem semantics reasoning process using monolithic intelligence techniques, adopting HMM or LDA algorithms, and HIT (i.e., the hybrid LDA-HMM algorithm). The discussed example concerns reasoning about semantics of a TCP-based file transfer service based on capturing raw network-data related to TCP-based services and learning data patterns. The following subsection shows DVA operation stage in NMemAgents to pave the way for SM reasoners to execute semantics reasoning.

Through a defined reasoning window, SM learns patterns of reduced-dimensional data profiles formed by DVA processes and maintained in the storage memory using the implemented ARL algorithm and FMF for recognizing and classifying attributes, respectively. SM recognizes the frequency of each data profile in the memory and its comprised attributes. For example, a captured profile might have multiple instances with the following attributes: P:={date, time, src_IP, dest_IP, packet_size, packet_type, service, protocol, port_number, sequence_number, MAC_address}. DVA adopts LSH to reduce the dimensionality of each profile. The profile P after applying LSH for selecting specific attributes and with a hash function length equals five will be P_LSH:={packet_size, packet_type, service, protocol, port_number}. Utilizing ARL and defined FMF for analyzing the data profile P_LSH gives the following: 10000 profile's instances with the attributes vector **Attr**: {$\mathbf{a^1}$= large_TCP-SYN_packet_size, $\mathbf{a^2}$= TCP-SYN_packets, $\mathbf{a^3}$= file_transfer_service, $\mathbf{a^4}$=abnormal_ port_number}.

We show in the next paragraphs the operation technique for three implemented algorithms (HMM, LDA and HIT) for semantics reasoning based on processes of data representation and dimensionality reduction which are discussed in the last paragraph. Due to adopting different algorithms for semantics reasoning, various output semantics will be formed and kept.

Firstly, HMM-based reasoning model is used to extract semantics regarding the behavior of TCP-based file transfer services. The input states to HMM are described as sequences. Each input state represents one learned and classified data-attribute. For

example, "*large TCP-SYN packet size*" is a learned and classified attribute based on captured values in the TCP *packet size* and *packet type fields* in a data profile. For HMM operation, we assume that we have four possible input states ST_1, ST_2, ST_3 and ST_4 to HMM for estimating the behavior of those services. Those input states represent the extracted and classified data attributes per each data profile. Those states are data attributes defined in the ***Attr*** vector. The estimated observation (or concept class) based on any sequence of the previously mentioned attributes is "**O**: *Abnormal TCP-based service*". HMM extracts that concept class based on its defined operation parameters. For instance, the observation probabilities B, using the maximum likelihood estimator, for the concept class **O** and the other concepts are high (e.g., 0.9) based on having any input attribute (i.e., state ST_i and $i \leq 4$) of the defined ***Attr*** attributes vector $(\mathbf{a^1,a^2,a^3,a^4})$. Also, the transition probability *A* between any two input features (ST_i and ST_j, and $i \neq j$) has the same value. This increases the probability of having the targeted **O** based on using the trained HMM with the forward algorithm over sequences of interesting attributes.

Secondly, LDA-based reasoning model is used to reason about semantics for TCP-based file transfer services, LDA after 1000 iterations assigns attributes in P_LSH to the following feature topics: "*Abnormal TCP-SYN packet size*", "*TCP-based File transfer service*", "*Abnormal TCP control packet*", "*Abnormal TCP-based file transfer service*". LDA samples the feature topics as the following example. The classified attribute $\mathbf{a^1}$, large_TCP-SYN_ packet_size, has high prior association weight β (e.g. $\beta > 0.75$) with the semantic topic, *Abnormal TCP-SYN packet size.* So, the updated attribute-topic posterior association weight φ for $\mathbf{a^1}$ after 1000 iterations will be high with respect to the assigned feature topic. In addition, the prior association weight (α) of P_LSH with the feature topic "*Abnormal TCP-based file transfer service*" is high (e.g., $\alpha > 0.8$). Then, LDA samples P_LSH that specific topic based on the updated topic-profile posterior association weight θ taking into consideration the number of profile's attributes assigned to that topic (e.g., two attributes) and the total number of profiles' attributes (e.g., P_LSH has four attributes).

Finally, the hybrid LDA-HMM algorithm is designed to produce more meaningful information from analyzed data profiles and to form richer associations amongst extracted semantics. The hybrid algorithm extracts latent features of each data attribute in analyzed data profiles relying on learned and classified attributes in analyzed data profiles in the storage memory. Then, the hybrid algorithm outputs data semantics for each data profile based on a sequence of extracted latent features of the whole data profile. The LDA-HMM algorithm looks at both a) data attributes and related latent features within data profiles and b) the sequence of extracted latent features within each data profile. The implemented LDA algorithm performs semantic of feature topics modeling for analyzed data profiles and their comprised attributes. The output of that modeling process is a set of syntax states per each profile that is fed to HMM to reason about semantics concerning each profile. Utilizing simple feature classification

techniques, e.g., using FMF, enables assignment of membership degrees for some extracted feature topics which have related designed FMF. These degrees enable SM to build and update DNCO. In the example of reasoning about TCP-based file transfer service semantics and using the same operation parameters of LDA in the previous semantics reasoning case, the latent feature topics ***T*** extracted by the LDA algorithm from P_LSH and its comprised attributes might be T:={ $\mathbf{t^1}$= Abnormal TCP-SYN packet size, $\mathbf{t^2}$= TCP-based File transfer service, $\mathbf{t^3}$= Abnormal TCP control packet,$\mathbf{t^4}$= Abnormal TCP-based file transfer service }. An input sequence comprised the previous latent feature topics (i.e., any sequence order of ($\mathbf{t^1,t^2,t^3,t^4}$)) to a designed HMM might yield the following semantics or observations : O:={ $\mathbf{O_1}$=File_Transfer_Service_ Behavior,$\mathbf{O_2}$=TCP_based_Service_Behavior,$\mathbf{O_3}$=Abormal_TCP_based_Service_Operatio n, $\mathbf{O_4}$=TCP_SYN_Flood_Attack}Accordingly, a simple DNCO can be built via the obtained observations or concept classes. For instance, the top parent class will be O_1 and it will have three child classes $\mathbf{O_2}$, $\mathbf{O_3}$ and $\mathbf{O_4}$.The concept class $\mathbf{O_3}$ will be a child class for the parent class $\mathbf{O_2}$ and so on. In addition, the four classified attributes in the ***Attr*** vector will be registered as FBS aspects for the lower child concept class $\mathbf{O_4}$.

Operation Discussion of and Qualitative Comparison among HMM-,LDA-, and LDA-HMM-based Models for Semantics Reasoning within NMemAgents

For learning network-semantics, intelligence techniques (ITs) for reasoning are used to extract meaningful information (e.g., ITs for predicting attacks and intrusions in networks [65, 76]) from raw data related to various Internet elements (e.g., services and protocols). NetMem adopts different techniques as discusses in previous subsections for reasoning about semantics and building ontology of network concepts. Semantics reasoning techniques learn patterns of raw data maintained in NMemAgents' storage memories. Those data are represented by DVA as data profiles of attribute-value pairs. Machine learning mechanisms [115], such as associative rule learning [116], besides Fuzzy logic [22] aid semantics reasoning operation by learning and classifying group of data attributes in maintained data profiles. The capability of implemented reasoning models in SM to reason efficiently about semantics is different where it depends on the abilities of related reasoning algorithms (see Figure 4.12). We show a qualitative comparison and differences between semantics reasoning using HMM, LDA, and LDA-HMM as follows:

- <u>HMM</u>
  - Output data semantics for each data profile based on a sequence of learned and classified attributes in the whole data profile *(HMM looks at the group and the sequence of data attributes within data profiles)*
    - *Example:* for TCP-based file transfer service, we might have the following attributes' sequence and output semantics
      - *Input sequence: Large packet size, normal port number, data packet, TCP protocol → the output: normal TCP-based service*

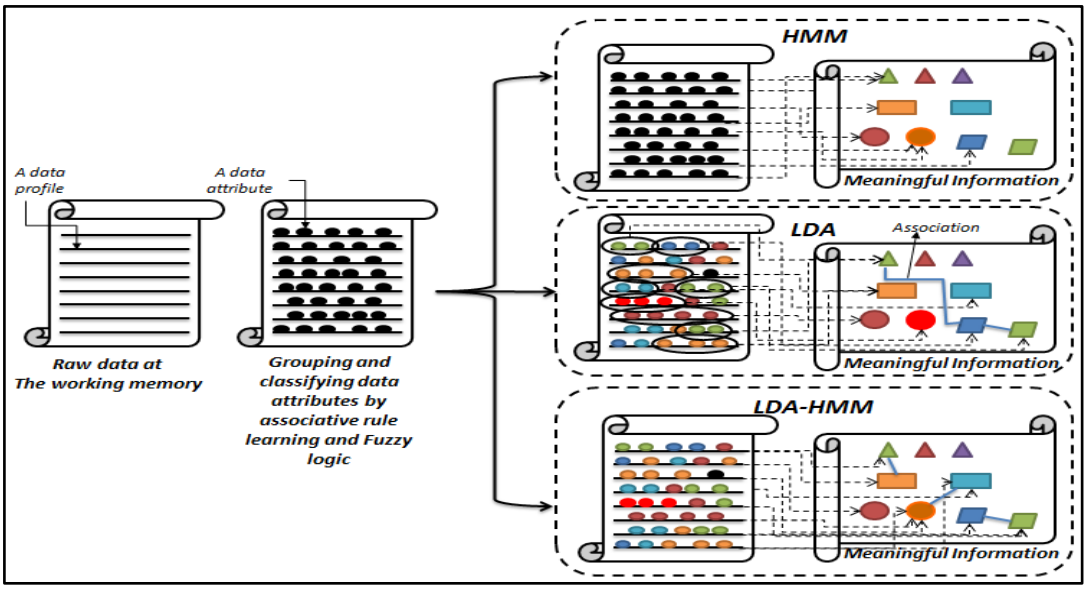


Figure 4.12 Semantics reasoning using different intelligence techniques

- LDA
  - Extract latent features of each data attribute in analyzed data profiles relied on learned and classified attributes in analyzed data profiles and then output data semantics based on extracted latent features within data profiles *(LDA looks at each data attribute and generates its related latent feature within each data profile. i.e., LDA makes semantic topic modeling for each data profile based on its comprised attributes)*
    - *Example:* for TCP-based file transfer service, we might have the following latent features and output semantics
      - *Extracted features: Large TCP packet size, File transfer service, normal TCP data packet, reliable protocol → the output: normal TCP-based File Transfer service*
- LDA-HMM
  - Extract latent features of each data attribute in analyzed data profiles relied on learned and classified attributes in analyzed data profiles and then output data semantics for each data profile based on a sequence of extracted latent features of the whole data profile *(LDA-HMM looks at both data attributes and related latent features within data profiles and the sequence of extracted latent features within each data profile)*
    - *Example: for TCP-based file transfer service, we might have the following latent features' sequence and output semantics*
      - *Input sequence: Large TCP packet size, File transfer service, normal TCP data packet, reliable protocol → the output normal TCP-based File Transfer service and normal behavior TCP operation*

4.5 Data/Semantics Storage/Update and Retrieval in/from NetMem

In this subsection, we show illustrations for processes of data/semantics storage/retrieval executed in each NMemAgent for achieving the goals of the overall NetMem system.

4.5.1 Data/Semantics Storage and Update

DVA component in NMemAgents located in the lowest NetMem level collects tons of multi-dimensional raw network-data through, for example, sensory system using remote terminal units [66] and/or management information bases (MIBs) [18] located at network entities. However, DVA does not store captured raw data as their original form and size in the storage memory. Some techniques for sketching data structure through LSH and data models are used to represent data as profiles of attribute-value pairs, and minimize required storage space and cost. Additionally, DVA at upper NMemAgents collect data and semantics with lower abstraction levels from NMemAgents at lower levels. DVA maintained data and semantics in the storage memory enabling SM processes for learning data patterns and extracting semantics with higher levels of abstraction. Recognized semantics are stored as associated concept classes in storage memories of NMemAgents at upper NetMem levels. NMemAgents can send data/semantics updates to upper agents when extracted semantics of their interest. Learned concepts are represented at different levels of granularity classifying concepts based on three network concerns; application, communication, and resource concern, and showing classes' FBS aspects. Maintained data and semantics can be accessed by hundreds of thousands of entities in different networking domains to learn, for instance, unfamiliar services, detection of abnormal flow, predict future events. Entities can only communicate with NMemAgents in the lowest NetMem level. Those agents can contact other agents in higher levels to retrieve required data/semantics.

4.5.1.1 NMemAgent Operation Flowchart

Figure 4.14 shows a sketch for data storage and semantics derivation processes in each NMemAgent. Figure 4.13 illustrates the overall framework of each NMemAgent

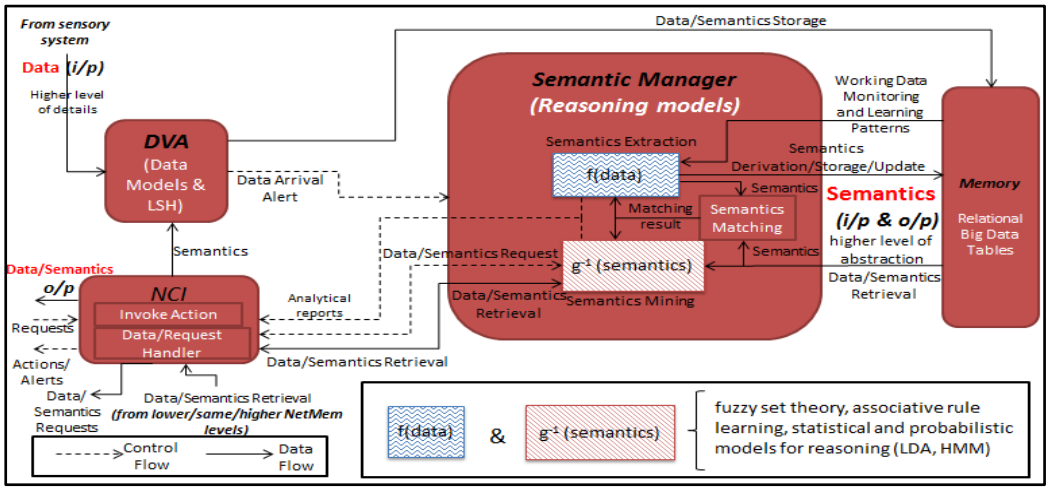


Figure 4.13 Data/semantics storage/retrieval processes in NMemAgent

clarifying its operations. Those operations show the processes beginning from capturing and storing raw data from different sources (e.g., Internet traffic and MIBs), reasoning about and keeping semantics with higher levels of abstraction and updating data/semantics in the storage memory until data/semantics requests and responses. The framework also exposes the capability of NMemAgents to interact together searching for required data/semantics and updating already maintained data/semantics in case of storage period expiration.

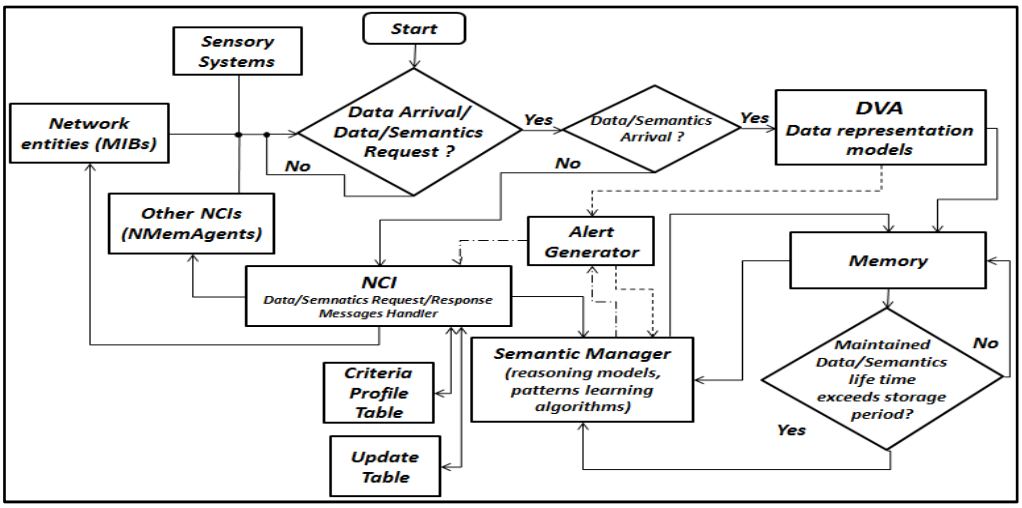


Figure 4.14 NMemAgent operation flowchart

We will discuss in the next subsections the processes of data/semantics derivation, storage and retrieval in NMemAgents and the overall NetMem system.

4.5.1.2 Semantics Derivation and Storage Processes in NMemAgents

Basically, DVA in each NMemAgent represents and stores data/semantics in agent's storage memory. SM monitors and learns patterns of data/semantics kept in the memory. Associative rule learning mechanisms are used to learn data attributes and features in different service profiles and concept classes, represented by DVA in relational big data tables in the storage memory. Statistical analysis techniques are applied for learning some data features like calculating number of each data profile and percentage of that feature existence in each data profile. Then, learned attributes and features are classified using Fuzzy set theory (Fuzzy membership functions). The classification process accepts continuous or discrete values and output discrete values based on possible states for each feature or attribute. After that, high level features extraction algorithms, semantics reasoning models and behavior estimation models will be used to reason about semantics and derive concept classes taking into consideration already maintained classes for updates. SM has a coordinator which a) receives alerts from DVA once data/semantics are stored; b) controls the operation of implemented features extraction and classification and reasoning models; and c) generate analytical reports to NCI to prepare actions and

alerts sent to network entities at certain cases like finding matching between learned of semantics a traffic flow and maintained concepts of an attack. Figure 4.15 shows the process of data storage and semantics reasoning process in NMemAgent. Table 4.3 discusses the main operations related to data storage and semantics reasoning processes in NMemAgent.

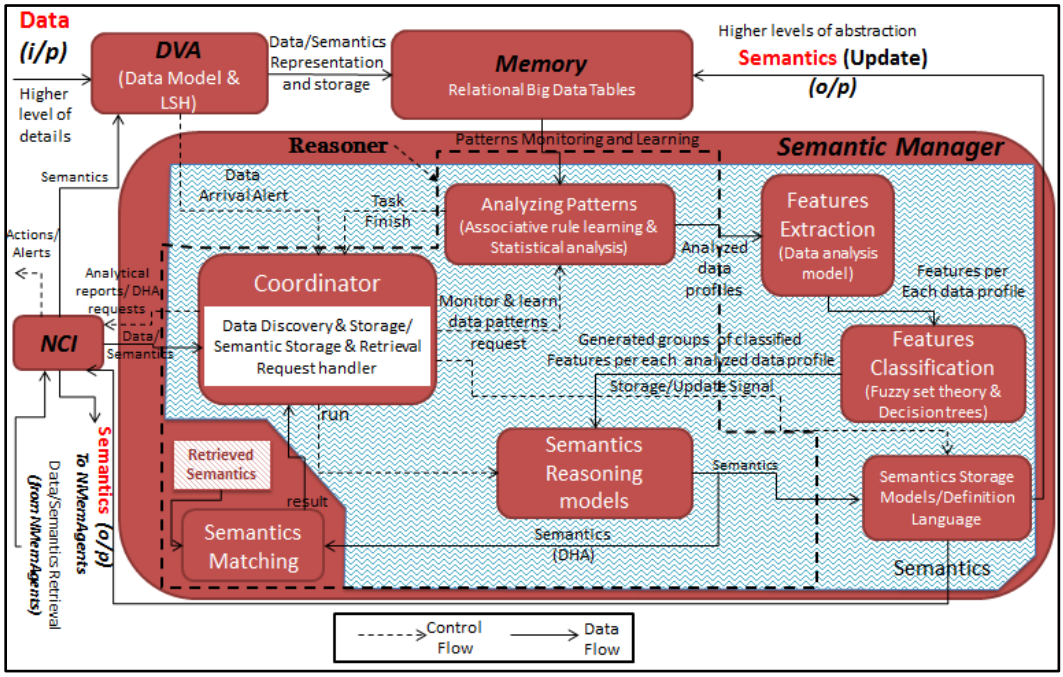


Figure 4.15 Data/semantics storage process in NMemAgent

Table 4.3 The operations of data/semantics storage and derivation in NMemAgent

Operation	Description
Data Collection	DVA receives: a) raw data from heterogeneous sources via a sensory system and/or MIBs; b) data/semantics via NCI from NMemAgents. DVA sends an alert to the semantic manager
Data Storage	DVA uses defined data models to store data/semantics in storage memory's big data table.
Data Patterns Learning	SM monitors and analyzes continuously patterns of maintained data/semantics for learning data attributes using associative rule learning and a statistical analysis model. SM uses Fuzzy set theory (defined FMF) for classifying discovered attributes.
Features Extraction	A data analysis process using data feature extraction algorithms is executed to discover and extract features for analyzed data profiles
Features Classification	Set of defined membership functions are used to classify features per each analyzed data profile
Semantic Reasoning	Extracted and classified features per each data profile is

	entering implemented reasoning models to generate data semantics, which are represented as associated concept classes
	SM uses statistical and probabilistic model to reason about semantics (i.e. most probable concepts) related to the input groups of data concerns of each analyzed data profile
Semantics Matching	Extracted concept classes are matched using a behavior estimation model with previously developed concepts in the storage memory to build/update ontology of concepts and to invoke some actions based on the matching result (e.g. attack alert generation, mismatched prediction, data ready for retrieval).
Semantics Storage	SM might update concept classes or semantics in the storage memory based on matching result and it sends learned semantics to NCI

Data/Semantics Construction Process within the overall NetMem system (from lower NetMem levels to upper levels):

- Agents at lower NetMem levels send data/semantics with lower levels of abstraction to agents at same/upper NetMem levels according to the following situations:
    1. Lower level agents have entries in their update tables concerning those data/semantics which are of interest to agents in same/upper levels.
    2. Lower level agents receive queries from agents in same/upper levels concerning specific data/semantics.
- Agents at higher NetMem levels send data/information queries to agents at lower levels in two cases:
    1. Agents can not meet demands of data/information required by agents at lower levels.
    2. Interesting data/semantics are invalid due to the expiration of their lifetimes.

Figure 4.16 shows the flowchart for the continual data/semantics update and storage processes in NetMem. The figure illustrates the various operations executed by NMemAgents at different NetMem system levels. For example, NMemAgents at low NetMem levels capture data/semantics from agents in previous levels and they extract semantics with higher levels of abstraction based on learning patterns of collected data/semantics. If those NMemAgents have update entries in their update tables concerning obtained semantics, they will send new updates to requesting agents. Also, NMemAgents investigate the expiration of storage duration and based on that they send other agents asking for new updates.

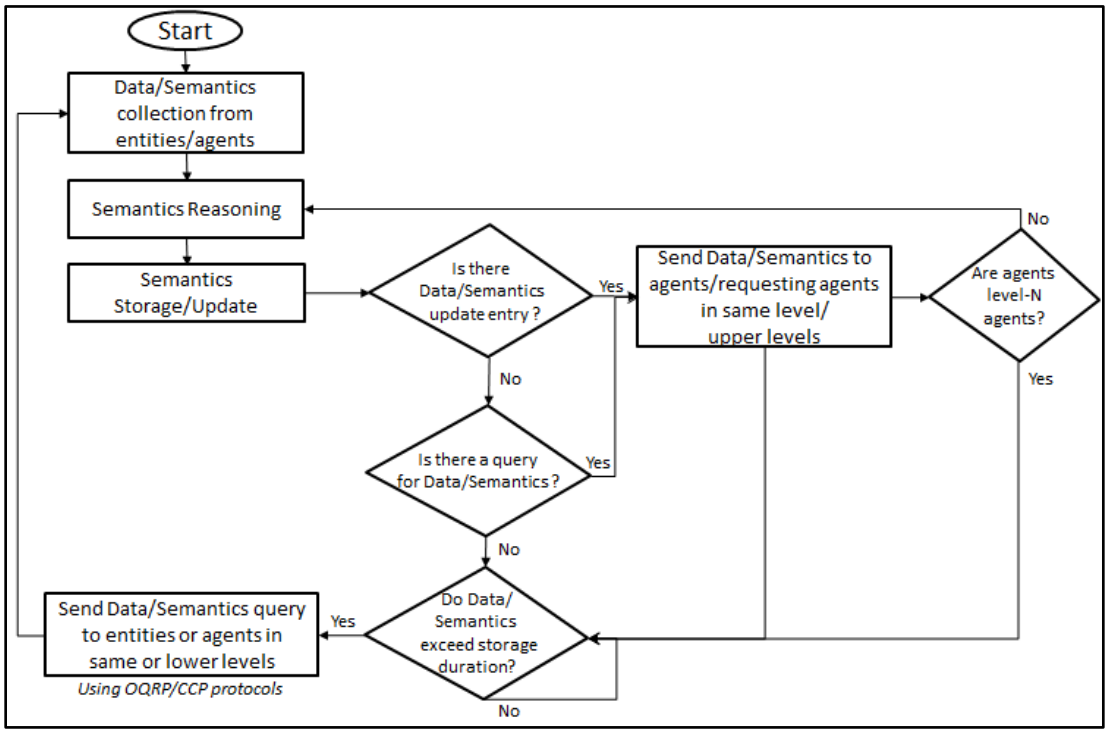


Figure 4.16 Data/information storage/update within NetMem

4.5.2 Data/Semantics Retrieval Process

Network entities can send NetMem data/semantics requests through NCI components in NMemAgents located at the lowest NetMem level. SM coordinator will send implemented statistical and probabilistic algorithms (i.e., reasoning model and behavior estimation models) to search for required data/semantics and to retrieve the most probable semantics. Relational tables of maintained data/semantics in the storage memory will be accessed to investigate the capability of meeting requests. Moreover, neighboring NMemAgents at same, lower and upper NetMem level can communicate asking for data/semantics of their interest. Once required data/semantics are found, NCI will get them and send responses including required information to network entities and agents. An alert of task finish will be sent to the coordinator declaring the completeness of the data/semantics retrieval process. Figure 4.17 and Table 4.4 shows the main operations of the data/semantics retrieval process in a NMemAgent.

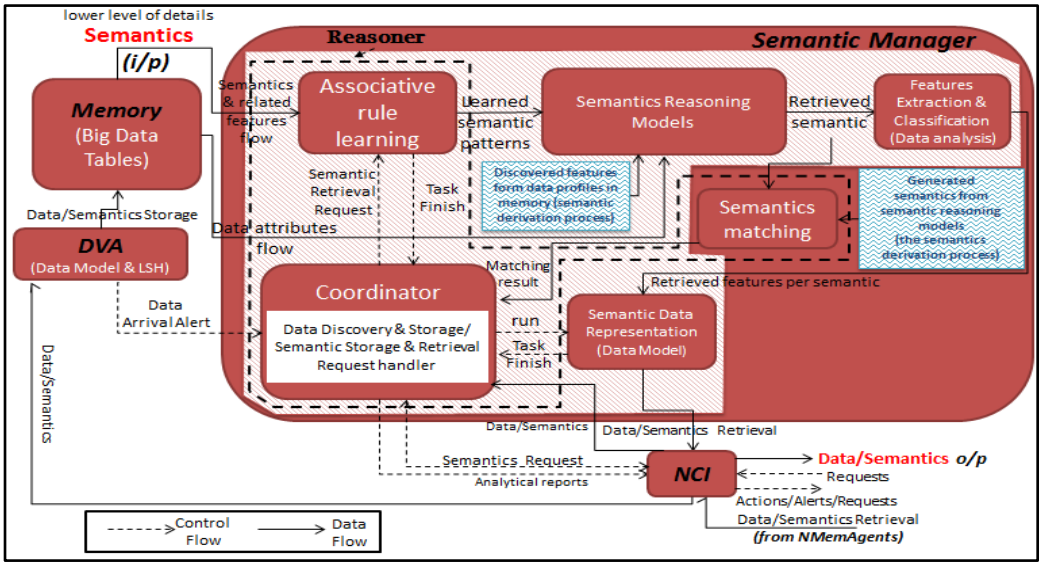


Figure 4.17 Data/semantics retrieval process in NMemAgent

Table 4.4 The operations of data/semantics retrieval in NMemAgent

Operation	Description
Semantics Retrieval	Based on signal from NCI, SM gets information about required semantics. Then, SM will run associative rule learning algorithm to learn already maintained concept classes and related features.
Semantics Extraction	SM runs the implemented reasoning models with the defined behavior estimation models to extract the most probable concept classes. SM might send the coordinator if it does not find similar information.
Features Extraction & Classification	Features of retrieved concept classes are extracted and classified using a data analysis model using FMF.
Semantics Representation	Based on extracted concept classes and related features, SM will represent required information using defined models and send information to the coordinator.
Semantics Retrieval	Once concept classes are ready, coordinator sends alert to NCI to send required information to requesting entities or agents.

4.5.2.1 Data/information Retrieval Process within the overall NetMem system (from longer-term NetMem levels to shorter-term levels)

- Data/semantics in NMemAgents located at lower NetMem levels are with lower levels of abstraction (i.e, higher levels of details) compared with semantics maintained in NMemAgents located at upper NetMem levels.
- NMemAgents located at same NetMem level will have same abstraction level of maintained data/semantics.

- Agents at lower NetMem levels send data/semantics queries to agents in higher NetMem levels.
- Agents at NetMem levels search for help from agents at upper levels in case that they can meet data/semantics requests from network entities/agents at lower levels.

Figure 4.18 shows the flowchart for the process of data/semantics retrieval from NetMem. The retrieval process considers requests sent by network entities in the network level or by NMemAgents at NetMem system levels. Once request receiving NMemAgents find required data/semantics, they will send information to requesting network entities or NMemAgents located at lower NetMem system levels.

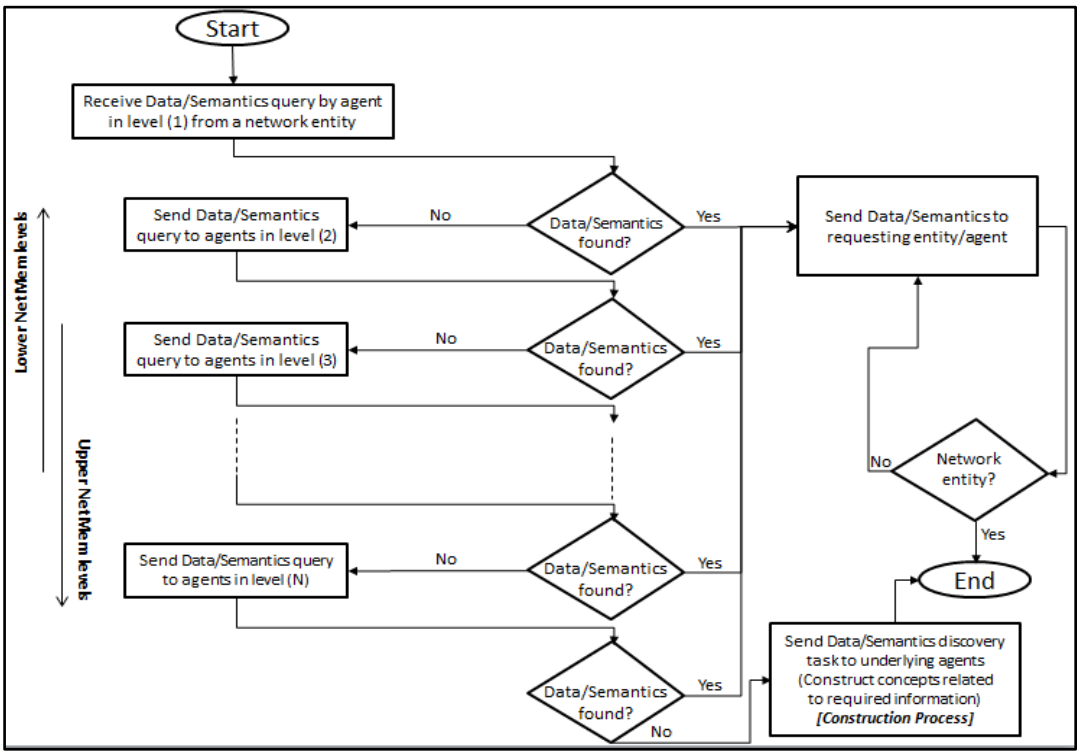


Figure 4.18 Data/semantics retrieval from NetMem

4.5.2.2 The Framework of Data/Semantics Request/Retrieval within NetMem

Figure 4.19 illustrates the processes of data/semantics request/retrieval processes in NetMem showing the interaction between two NMemAgents in two different levels and among the various NMemAgent's components. The figure depicts a framework for the NetMem system where it shows some NMemAgents operation cases mentioned in last paragraphs. For instance, the figure illustrates two data/semantics request cases between NMemAgents showing the interaction amongst NMemAgent's comprised components (NCI, DVA, SM and storage memory). The first case is regarding QoS information (i.e., data with lower levels of abstraction) about a specific service. The requested information

class concerns an application network concern (i.e., a file transfer service). The NCI of the receiving NMemAgent at the lowest NetMem level asks SM to retrieve the required information. The second case shows a behavior concept class (i.e., semantics with higher levels of abstraction) requested by a NMemAgent concerning the normal behavior of TCP (a communication network concern) in wireless environments. NMemAgents in the lowest NetMem level are not able to meet the information request. So, agents communicate with other agents at the next upper NetMem level for retrieving required semantics.

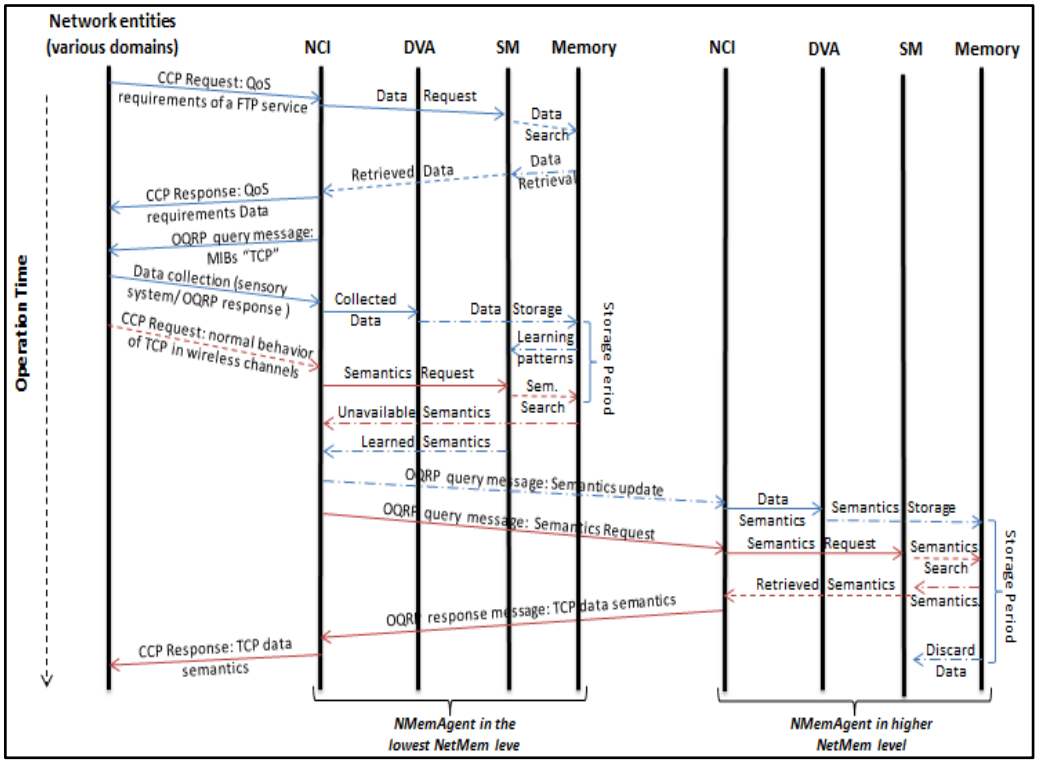


Figure 4.19 The framework of data/semantics request/retrieval within NetMem

4.6 Conclusion

In this chapter, we discussed semantics management in NetMem using monolithic and hybrid intelligence techniques for extracting semantics and building dynamic ontology of network concepts. The hybridization of more than one monolithic intelligence technique can help in integrating the capabilities of each technique when it is used separately. Additionally, we discussed the operations of semantics storage, update and retrieval to/from NetMem.

Chapter 5

5 NETMEM FORMALIZATION, ANALYSIS AND OPTIMIZATION

5.1 Overview

We propose an analytical model for depicting the design and evaluation of the efficiency of NetMem as a distributed intelligent multi-agent system. We analytically evaluate NetMem efficiency in case of data retrieval and abstraction processes. The model was designed using mathematical equations and relationships which relying on queuing theory. Furthermore, an optimization problem was addressed for NetMem configuration in order to improve NetMem efficiency. We utilized the genetic algorithm (GA) [117] to solve the formulated optimization problem. The output provided recommended semantics reasoning models to be implemented in NMemAgents located at different NetMem levels. The following present the formal representation model, the analytical model and the optimization of NetMem.

5.2 NetMem Formal Representation Model

NetMem can be defined as follows. [The symbol ($^{+}$) attached to some terms in the description of NetMem indicates that the related term should have at least a value of one.]

- **Structure**

NetMem is modeled as NMemSys: {AgentComponent, DataComponent, Agent_DataMapping} where:

A. AgentComponent: defines the basic building unit, or NMemAgent, located at hierarchy of levels in NMemSys. AgentComponent is described as: {NMemAgent, AgentLevelHierarchy}, where:
   1) NMemAgent:={R, V, D, M, W, Cp, Ct}, where:
   - R: set of intelligence techniques implemented in NMemAgent to learn data patterns, reason about semantics and build concept ontology.
   - V: defines the data virtualization technique used in NMemAgent located at the first NetMem level.
   - D: defines the dimensionality reduction algorithm utilized in NMemAgent located at the first level.
   - M: defines the representation model adopted by NMemAgent to represent data and semantics.
   - W: defines the implemented scalable storage technique for realization the memory unit inside NMemAgent for maintaining data/semantics.

- Cp: set of communication protocols that are defined in NMemAgent to facilitate: i) interaction among agents and between agents and Internet elements for data/semantics requests and responses; and ii) cooperation among various operations inside each agent.
- Ct: defines the controller unit in NMemAgent which is responsible for managing operations inside the agent, such as selecting the appropriate intelligence technique for semantics reasoning.

2) AgentLevelHierarchy: defines the hierarchical structure of NMemSys in terms of sets of NMemAgents (i.e., more than one NMemAgent) located at a hierarchy of levels. AgentLevelHierarchy can be described as:

$$\text{AgentLevelHierarchy}:=\{N^{+}, G, L\}, \text{ where:}$$

- N: defines the number of NMemSys levels.
- G: defines the distribution of NMemAgents over a hierarchy of N NMemSys levels. G can be described as:

  $G:=(S_1^{+}, S_2^{+}, \ldots.. , S_i^{+}, \ldots\ldots, S_{N-1}^{+}, S_N^{+})$, where

  S_i: defines the set of NMemAgents per NMemSys level i where $(1 \leq i \leq N)$ and $(S_1>\ldots>S_i>S_{i+1}>\ldots>S_N)$. Total number S of NMemAgents in NMemSys equals $\sum_{i=1}^{N} S_i$.
- L: defines the possible operation interaction between NMemSys levels where S_i NMemAgents at level i can interact with S_{i-1} agents located at the lower level i-1 and S_{i+1} agents at the upper level i+1.

B. DataComponent: defines data/semantics and ontology of concepts that can be maintained at NMemSys. DataComponent can be described as:

$$\text{DataComponent}:=\{\text{RawData, Semantics}\}$$

1) RawData: describes raw data that are collected from different sources, such as Internet traffic and offline databases. RawData can be described as:

$$\text{RawData}:=\{A, P, Cn, I_{e,raw}\}, \text{ where:}$$

- A: set of data attributes.
- P: defines raw data profiles where each profile comprises attribute-value pairs.
- Cn: set of network concerns on which data attributes and data profiles will be classified. Cn is defined as Cn:={ApplicationConcern, CommunicationConcern, ResourceConcern)
- $I_{e,raw}$: sets of Internet elements which are defined by raw data profiles and related attributes.

2) Semantics: describes data with higher levels of abstraction and more meaningful information. Semantics can be formally represented by ontology of correlated concept classes at different levels of granularity. Semantics can be described as ontology O where:

$$O:=\{n_c, Cn, I_{e,c}, I_c^{c}, M_c^{T}, FBS_c, I_n\}, \text{ where}$$

- n_c: set of concept classes that can be identified via O.

- Cn: set of network concerns (as defined in RawData) on which discovered concept classes will be classified.
- $I_{e,c}$: sets of Internet elements which have defined concept classes in O.
- I_c^c: a matrix, Figure 5.1(a), for identifying types of n_c concept classes according to ACR concern. A concept class will have one entry which refers to the concern type of the class.
- M_c^T: a taxonomy matrix, Figure 5.1(c), which maintains concept classes and their related features mentioning their membership degrees and values.
- FBS_c: a string matrix, Figure 5.1(b), which contains functional, behavioral and structural aspects of each concept class related to an Internet element in I_e. Entries in that matrix are extracted and induced based on features per each concept class in M_c^T matrix.

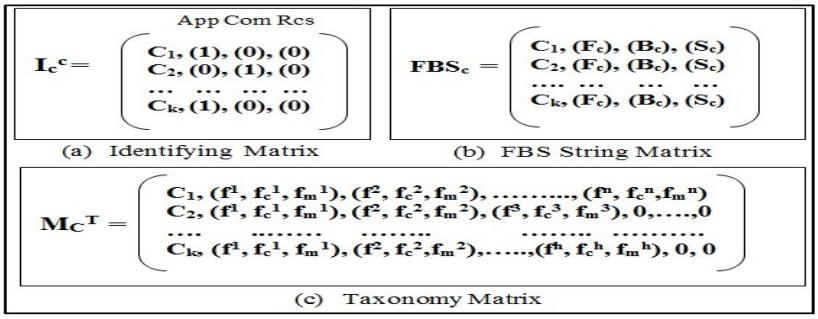


Figure 5.1 Ontology matrixes

- I_n: defines the inheritance among identified concept classes in O. It shows lower level or abstracted concept classes with sub-class relationships to other parent or composed classes (i.e., upper level concept classes). In can be described as:
  I_n:={{C_U},{C_L}}={{C_1,C_2,….., C_x},{($C_{l1,1}$, $C_{l2,1}$,…..),($C_{l1,2}$, $C_{l2,2}$,…..),……,($C_{l1,x}$, $C_{l2,x}$,…)}}, where
  C_U: defines set of upper level or parent concept classes where C_U:= {C_1,C_2,….., C_x} and C_L: defines sets of lower level or child concept classes for each concept class in C_U.

C. Agent_DataMapping: describes the interrelation between AgentComponent and DataComponent at NMemSys. That interrelation depends on the hierarchical structure of NMemSys which is presented in terms of set of levels where each level contains NMemAgents. NMemAgents at a NMemSys level maintain data/semantics with same abstraction level. The sets of NMemAgents located at multi-levels in NMemSys provide ontology hierarchy with identified and correlated concept classes at different levels of abstraction. Agent_DataMapping is described as:

Agent_DataMapping:={AgentData, OntologyHierarchy}, where

1) AgentData: defines collected/retrieved and abstracted data in NMemAgents at the NMemSys to form their ontologies of identified concept classes. AgentData can be described as: AgentData:={C, O_a}, where:

- C: defines maintained raw data/semantics at NMemAgents. C can be described as:

$C:=\{C_{RawData}, C_{Semantics}\}$, where:

$C_{RawData}$: defines maintained raw data in NMemAgents at the lowest NMemSys level as data profiles of attribute-value pairs.

$C_{Semantics}$: defines maintained semantics in NMemAgents as concept classes as specific level of abstraction. $C_{Semantics}$ can be described as:

$C_{Semantics}:=\{$AbstractedConcept, ComposedConcept$\}$, where:

AbstractedConcept: defines set of concept classes that give valuable information about a specific ACR concern. Those classes are low level concept classes where they represent sub-class of relationships (i.e., child class) to composed or parent concept classes at specific level of abstraction.

ComposedConcept: define set of concept classes that give at specific level of abstraction correlated information about more than one ACR concern. Those classes represent upper level or parent concept classes to more than one abstracted or child class.

- O_a: defines the built ontology O with correlated concept classes defined in $C_{Semantics}$ at a specific level of data abstraction at each NMemAgent located at a certain NMemSys level. Concept classes are represented related to ACR concern using defined models in M.

2) OntologyHierarchy: defines the formed hierarchy of concept classes with various abstraction levels in the hierarchical structure of NMemSys according to built ontologies at NMemAgents located at N levels. OntologyHierarchy can be described as: OntologyHierarchy:=$\{n_{c,i}, LoA_i, T_i\}$, where:

- $n_{c,i}$: defines the number of identified concept classes at level i. Each NMemAgent at level i will have n_c identified concept classes in its formed ontology $O_{a,i}$. There is a probability for having similar concept classes among NMemAgents at level i. So, $n_{c,i}$ expresses about the summation of numbers of distinct identified concept classes at all NMemAgents at level i and ($n_{c,1}<n_{c,2}<\ldots < n_{c,i} <\ldots<n_{c,N-1} < n_{c,N}$).
- LoA_i: describes level of abstraction for maintained concept classes at NMemAgents at level i. Kept classes related to specific Internet elements at level i are of: i) lower abstraction level compared with the one of classes related to same elements at upper NMemSys levels (i.e., levels: i+1,…….., N-1, N), and ii) higher abstraction level compared with the one of kept classes at lower NMemSys levels (i.e., levels: 1, 2, ….., i-1). Concept classes with the lowest abstraction level are maintained in agents at NMemSys level 1, while, classes with the highest abstraction level are kept at agents at level N.
- T_i: defines the storage duration of $n_{c,i}$ identified concept classes at level i where ($T_1 < T_2 <\ldots< T_i < <\ldots< T_{N-1} < T_N$). Maintained concept classes at the highest NMemSys level (i.e., level N) will have the longest storage duration compared with kept classes at lower NMemSys levels.

- **Function**

NMemSys function can be defined and described according to the function of each comprised NMemAgent at all NMemSys levels. The function of NMemAgents, or NMemAgentFunction, can be defined as follows:

NMemAgentFunction:={ NMemSysD/S, Task, Connection, DataFlow} where:

A. NMemSysD/S: represents data input/output to/from NMemAgents at NMemSys. NMemSysD/S can be described as:

$$NMemSysD/S := \{Data_{Low\_LoA}, Data_{High\_LoA}\}, \text{ where}$$

$Data_{Low\_LoA}$ is input/output data/semantics with lower levels of abstraction to/from agent, $Data_{High\_LoA}$ is input/output semantics with higher levels of abstraction to/from agent.

B. Task: set of functions that NMemAgent will perform. It can be described as follows:

$Task:= \{f_d(data,Dim), f_a(NMemSysD/S,R), f_r(NMemSysD/S), f_m(NMemSysD/S), f_l(NMemSysD/S,R), f_u(NMemSysD/S, M), f_s(NMemSysD/S, M)\}$, where

- f_d: function of dimensionality reduction over data using the algorithm defined in D (in case of agents in the first NMemSys level).
- f_a: function of abstracting NMemSysD/S to reason about data with higher level of abstraction using a reasoning model defined in R.
- f_r: function of retrieving NMemSysD/S of interest.
- f_m: function of matching new discovered data with already maintained NMemSysD/S for learning novel information and updating built ontology.
- f_l : function of discovering and classifying data attributes or features.
- f_u: function of updating NMemSysD/S in agent's storage memory using a defined representation model in M according to received data/semantics updates from other agents.
- f_s: function of storing NMemSysD/S in agent's storage memory using a defined representation model in M.

C. Connection: set of available communication channels among NMemAgents in various levels.

$$Connection:= \{c_{i,i}, c_{i,i+1}, c_{i,i-1}, c_{i,e}\}, \text{ where}$$

- $c_{i,i}$: set of bidirectional communication channels among agents at level i.
- $c_{i,i+1}$: set of bidirectional communication channels between agents at level i and agents at the next upper level i+1
- $c_{i,i-1}$: set of bidirectional communication channel between agents at level i and agents at the underneath level i-1
- In case of agents at the lowest level, $c_{i,e}$: a bidirectional communication channel between agents and network entities/data sources in various network domains.

D. DataFlow: represents different data flows for data abstraction and retrieval among NMemAgents.

$$DataFlow:= \{flow(c_{i,i+1}), flow(c_{i,i}), flow(c_{i,i-1}), flow(c_{1,e})\}, \text{ where}$$

- flow($c_{i,i+1}$): a) data abstraction process →flow of NMemSysD/S from agents at level i to agents at level i+1, and b) data retrieval process → flow of NMemSysD/S from agents at level i+1 to agents at level i.
- flow($c_{i,i}$):flow of NMemSysD/S updates and retrieval between agents at level i.
- flow($c_{i,i-1}$): a) data retrieval process →flow of NMemSysD/S from agents at level i to agents at level i-1, and b) data abstraction process → flow of NMemSysD/S from agents at level i-1 to agents at level i.
- flow($c_{1,e}$): a) data abstraction process →flow of data from entities/data sources in heterogeneous network domains to agents at the lowest level, and b) data retrieval process → flow of NMemSysD/S from agents at the lowest level to network entities at the network level.

- **Behavior**

NMemSys behavior can be described according to the behavior of comprised NMemAgents at N NMemSys levels. At any instant of NMemSys operation time, the behavior B of NMemSys is given by:

B(***b***):= {E, I}, where

- E: set of events that can describe behavior index ***b*** of NMemAgents located at N levels.
- I: set of agent-based measurable indicators whose values (real or integer values) refer to ***b***.

Different behavior indexes are defined with various generated events to study the dynamic behavior of NMemAgents located at multi-levels. The following behavior indexes are adopted: i) Memory Type (**MT**), ii) Pattern Change Rate (**PCR**) and iii) Data Level of Abstraction (**DLA**). The following points describe how NMemSys behavior can be assigned and studied according to the previously mentioned behavior indexes.

1) B(**MT**): indicates NMemSys behavior in terms of memory behavior at NMemAgents located at N levels. B(**MT**) can be described as follows:

B(**MT**):={E, I}={(e_1, e_2,……., e_N), (TTL_1, TTL_2,……, TTL_N)}, where:

- e_i describes the memory type of NMemAgents at level i.
- TTL_i: refers to the time to live value of data/semantics (i.e., memory content) in agents at level i.
- Event transitions: this shows the different analyzed behavior according to movement from a NMemSys level to another level.
  1) {e_i, TTL_i} to {e_{i-1}, TTL_{i-1}}: move from NMemAgents at level i with shorter-term memories to agents at level i-1 with longer-term memories where $TTL_i > TTL_{i-1}$.

2) $\{e_{i-1}, TTL_{i-1}\}$ to $\{e_i, TTL_i\}$: move from NMemAgents at level i-1 with longer-term memories to agents at level i with shorter-term memories where $TTL_{i-1} < TTL_i$.

- The events from e_1 to e_N represent NMemSys memory type in each comprised level which varies from the shortest-term storage at NMemAgents at level 1 to the longest-term storage at agents at level N. Events change in NMemSys refers transition from a level to another level with different storage durations for memory contents at NMemAgents.

2) B(**PCR**): indicates NMemSys behavior in terms of rate of data/semantics pattern change at NMemAgents located at N levels. B(**PCR**) can be described as follows:

B(**PCR**):={E, I}={(e_1, e_2,……., e_N), (Ws_1, Ws_2,……, Ws_N)}, where:

- e_i describes change rate of data/semantics patterns at NMemAgents at level i
- Ws_i: refers to reasoning window size adjusted by semantics manager in NMemAgents at level i. Small window sizes at agents mean high pattern change rate of feeding data/semantics.
- Event transitions: this shows the different analyzed behavior according to movement from a NMemSys level to another level.
  1) $\{e_i, Ws_i\}$ to $\{e_{i-1}, Ws_{i-1}\}$: move from NMemAgents at level i with lower pattern change rate to agents at level i-1 with higher pattern change rate where $Ws_i > Ws_{i-1}$.
  2) $\{e_{i-1}, Ws_{i-1}\}$ to $\{e_i, Ws_i\}$: move from NMemAgents at level i-1 with higher pattern change rate to agents at level i with lower pattern change rate where $Ws_{i-1} < Ws_i$.
- The events from e_1 to e_N represent change rate of data/semantics pattern at NMemSys in each comprised level which varies from the highest pattern change rate at agents at level 1 to the lowest pattern change rate at agents at level N. Events change in NMemSys refers to transition from a level to another level with different data/semantics pattern change rates.

3) B(**DLA**): indicates NMemSys behavior in terms of abstraction level of maintained data/semantics in NMemAgents located at N levels. B(**DLA**) can be described as follows:

B(**DLA**):={E, I}={(e_1, e_2,……., e_N), (Ms_1, Ms_2,……, Ms_N)}, where:

- e_i describes level of data abstraction at NMemAgents at level i
- Ms_i: refers to memory size of NMemAgents at level i. Small memory sizes at agents mean high levels of data abstraction where less detail can be found for maintained data.
- Event transitions: this shows the different analyzed behavior according to movement from a NMemSys level to another level.

1) $\{e_i, Ms_i\}$ to $\{e_{i-1}, Ms_{i-1}\}$: move from NMemAgents at level i with lower level of data abstraction to agents at level i-1 with higher level of abstraction where $Ms_i < Ms_{i-1}$.
2) $\{e_{i-1}, Ms_{i-1}\}$ to $\{e_i, Ms_i\}$: move from NMemAgents at level i-1 with lower level of data abstraction to agents at level i with higher level of data abstraction where $Ms_{i-1} > Ms_i$.

The events from e_1 to e_N represent data abstraction level at NMemSys in each comprised level which varies from the lowest abstraction at agents at level 1 to the highest abstraction level at agents at level N. Events change in NMemSys refers to transition from a level to another level with different data abstraction levels.

5.3 Analytical Model for NetMem

We provide an analytical model which is used to calculate NetMem overhead and response time in case of retrieving data classes. Additionally, the model calculates NetMem processing time in case of abstracting data semantics or concept classes.

5.3.1 The Network Model

A NetMem-based network comprises network entities and NetMem which comprises a hierarchy of NMemAgents located over multiple NetMem levels. Network entities can communicate only with NMemAgents located in the first NetMem level to retrieve data. NMemAgents can interact with agents at the same or the upper level to retrieve data. On the other hand, NMemAgents can communicate with agents at the same or the lower level to abstract data semantics and forming DNCO at this level of detail. The adopted network model is described as follows:

- N: number of NetMem system levels where NMemAgents are located.
- S_i: number of NMemAgents per NetMem system level i.
- n_s: total number of NMemAgents considering the total N levels = $\sum_{i=1}^{N} S_i$
- n_e: total number of network entities (e.g., hosts, gateways,….,etc.)
- n : total number of network entities and NMemAgents = $n_e + n_s$.
- $n_{c,i}$: number of identified concept classes at level i.
- To have a connected routing graph with high probability [118], the average degree ***g*** of agent/entity grows with O(log *g*). So, we assume that in case of data/semantics retrieval and semantics abstraction there will be a number of data units (DU) or messages transferred among S_i agents grows with order related to surrounding neighboring agents whether at the same level (i), the lower level (i-1) and the upper level (i+1).

5.3.2 Data Retrieval Process

5.3.2.1NetMem Overhead

- For NMemAgents in any level i (except i=1): number of transferred DU will grow with O $\log(S_i + S_{i+1}) = A_i D_i \log(S_i + S_{i+1})$ where:

- A_i: data sources' prediction accuracy that represents the percentage of data sources which have targeted classes and/or classes' requests and can send them to agent at level (i), and
- D_i: average number of generated DU and requests per class from data sources to agents at level (i). For S_i NMemAgents at level i, there are the following chances to have received/sent/passed DU through the routing graph among the different levels (i, i+1):
  - DU from S_{i+1} to S_i related to requests sent from and/or interesting updates by S_i
  - DU from S_i to S_i related to requests sent from and/or interesting updates by S_i
  - DU from S_{i+1} to S_{i+1} based on requests from S_i and/or interesting updates by S_i and S_{i+1}

- For NMemAgents in the first NetMem system level: number of transferred DU will approximately grow with O $\log(n_e + S_1 + S_2) = A_1 D_1 \log(n_e + S_1 + S_2)$ where there are collected DU and sent data requests from n_e network entities.
- T_c : the maximum storage time period per data/semantics (D/S) class entry in update tables in NMemAgents within NetMem System levels. (N.B. T_c value varies related to each D/S class and the NMemAgent where it is defined).
- A D/S class exists at level (i) with probability P_c due to the abstraction level of D/S and the implemented reasoning model. This value affects the storage of each D/S class in each NetMem level. We assume that high P_c (higher level of abstraction and D/S will be located in higher levels) means that much time is needed to extract D/S. Accordingly, long storage duration T_c will be needed to get that D/S class.
- Assume a D/S class is updated within $P_c T_c$ with a probability calculated from the exponential distribution: : $\int_0^{P_c T_c} {}^1/_\beta \; e^{-t/\beta} \, dt = 1 - e^{-P_c T_c/\beta}$ (5.1)

  Where β is average value of a D/S class survival (in time unit) in an update table

  $\beta = \left(\sum_{c=1}^{n_c} P_c \, T_c\right)/n_c$ (5.2)
- The average number of D/S class updates happened in the update table of a NMemAgent equals $n_c \times (1 - e^{-P_c T_c/\beta})$ (5.3)
- NetMem system overhead (O_{NetMem}): O_{NetMem} is defined as the expected number of query/update messages forwarded by NMemAgents plus number of data packets (query messages) related to specific D/S classes and collected by first level NMemAgents where the probability to collect a data packet related to a D/S class from every network entity is assumed that obeys Poisson distribution with mean equals λ packets that the probability that no collected packets with required D/S class through the storage duration of the class equals $e^{-\lambda_c P_c T_c}$

- $O_{NetMem} = \sum_{i=2}^{N}(A_i D_i \log(S_i + S_{i+1})(\sum_{c=1}^{n_{c,i}}(1 - e^{-\lambda_{c,i} P_{c,i} T_{c,i}})(1 - e^{-P_{c,i} T_{c,i}/\beta_i}))) + A_1 D_1 \log(n_e + S_1 + S_2)(\sum_{c=1}^{n_{c,i}}(1 - e^{-\lambda_{c,1} P_{c,1} T_{c,1}})(1 - e^{-P_{c,1} T_{c,1}/\beta_1}))$ (5.4)

Where:

- $A_i D_i \log(S_i + S_{i+1})$: represents the number of DU which are related to required classes and sent by possible data sources (or NMemAgents) in the same level and the upper level.
- $\sum_{c=1}^{n_{c,i}}(1 - e^{-\lambda_{c,i} P_{c,i} T_{c,i}})(1 - e^{-P_{c,i} T_{c,i}/\beta_i})$: is the total probability of having non-zero generated data units related to the targeted $n_{c,i}$ D/S classes within the storage duration of each concept class at level (i) agents.

5.3.2.2 NetMem Response Time

We assume, using the queuing theory [119], that each NMemAgent within a NetMem system level (i) is represented by M/M/1 queuing where one agent (considered as a server) has infinite buffer size with exponential distribution of inter-arrival and service times for messages received from other NMemAgents at same, upper or lower levels and from network entities (in case of NMemAgents at the first NetMem level)

- Using the M/M/1 queue discipline:
  - NetMem response time (R) for a query message

    $R = s + w$; (5.5)

    where s: service time which obeys exponential distribution with mean service time = 1/ μ where μ: mean service rate (messages/time unit) and w: is the waiting time which obeys exponential distribution with mean inter-arrival time = 1/λ where λ: mean generation rate (messages/time unit).
  - The average response time r per message processed by NMemAgent

    $r = 1/(\mu - \lambda)$ (5.6)
  - In order to not having saturated NetMem system with unbounded r_i response time at level (i), we assume that the load is equally distributed across S_i NMemAgents at level (i) with rate λ_i / S_i that $r_i = 1/(\mu_i - \lambda_i/S_i)$ where
    - λ_i: data generation rate from all possible data sources and related to various classes to agents at level (i).
    - S_i: number of agents at level (i).
    - λ_i/S_i: data arrival rate to an agent at level (i) assuming that the expected generated data will be distributed over S_i agents at level (i).
    - μ_i: the request service rate at an agent at NetMem system level (i) (we consider a parallel processing in all agents at level (i), So, μ_i refers to the request service rate at level (i))

    Note: r_i represents the response time of level (i) in case of having parallel processing in all agents at level (i).

- μ_i= 1/(mean service duration S_D) and S_D is the average of identical and independent random variables (message service duration of each possible defined D/S class $\boldsymbol{c}$ of n_c concept classes per NMemAgent).
    - So, $S_D = \frac{1}{\mu} = \left(\sum_{c=1}^{n_c} P_c\, T_c\right)/n_c$ (5.7)
    - where the service duration for each D/S class message is assumed to have a maximum value equals D/S class storage time T_c and its value affected by the probability of D/S class existence P_c in NetMem system based on the used reasoning model. Each model has its time complexity and this affects the duration of classes of data semantics in NetMem.
    - λ_i is assumed to equal generation rate of D/S messages that can be received by a NMemAgent within mean inter-arrival time equals $\left(\sum_{c=1}^{n_c} P_c\, T_c\right)/n_c$
    - For each NetMem system level (i):

$\lambda_i = (A_i D_i \log(S_i + S_{i+1})(\sum_{c=1}^{n_{c,i}}\left(1 - e^{-\lambda_{c,i} P_{c,i}\, T_{c,i}}\right)(1 - e^{-P_{c,i}\, T_{c,i}/\beta_i}))/((\sum_{c=1}^{n_{c,i}} P_{c,i}\, T_{c,i})/n_{c,i})$ (5.8)

where

- $A_i D_i \log(S_i + S_{i+1})$: represents the number of data units or messages which are related to required classes and sent by possible data sources (NMemAgents) in the same level and the upper level
- $\left(1 - e^{-\lambda_{c,i} P_{c,i}\, T_{c,i}}\right)$: the probability of having non-zero generated data units related to a certain class and sent to level (i) agents where $\lambda_{c,i}$ represents the arrival rate of data units related to a certain class through the storage duration of that class and $\lambda_{c,i} P_{c,i}\, T_{c,i}$ is the expected number of arrived class data units through the storage duration of that class
- $(1 - e^{-P_{c,i}\, T_{c,i}/\beta_i})$: the probability that a data unit related to a required concept class will be sent through the storage duration of that class at level (i) agents and β_i is the average storage duration or survival D/S class time at level (i) = $\left(\sum_{c=1}^{n_{c,i}} P_{c,i}\, T_{c,i}\right)/n_{c,i}$ where $P_{c,i}$ is the probability of a D/S class existence at level *i* at using a semantics reasoning model.

- Now, the analytical model to predict NetMem system response time of level *i* can be calculated by

$r_i = 1/(\mu_i - \lambda_i/S_i) = 1/(\left(\sum_{c=1}^{n_{c,i}} \frac{1}{P_{c,i} T_{c,i}}\right)/n_{c,i} -$

$(A_i D_i log(S_i + S_{i+1})(\sum_{c=1}^{n_{c,i}}\left(1 - e^{-\lambda_{c,i} P_{c,i}\, T_{c,i}}\right)(1 - e^{-P_{c,i}\, T_{c,i}/\beta_i}))/(\sum_{c=1}^{n_{c,i}} P_{c,i}\, T_{c,i})/n_{c,i})/S_i)$ (5.9)

- For N NetMem system levels the total response time $r = \sum_{i=1}^{N} r_i$
- The estimated response time r_e of NetMem system in case of feedback connections (information rehearsal):

$$r_e = \sum_{i=1}^{N} r_i =$$
$$1\left(\left(\sum_{c=1}^{n_{c,N}} \frac{1}{P_{c,N}\, T_{c,N}}\right) / n_{c,N} - (\mathrm{A_N D_N} log(S_N)\left(\sum_{c=1}^{n_{c,N}}\left(1 - e^{-\lambda_{c,N} P_{c,N}\, T_{c,N}}\right)\left(1 - e^{-P_{c,N}\, T_{c,N} / \beta_N}\right)\right) / \left(\sum_{c=1}^{n_{c,N}} P_{c,N}\, T_{c,N}\right)/n_{c,N}\right)/S_N) + \sum_{i=2}^{N-1}\left(1/\left(\left(\sum_{c=1}^{n_{c,i}} \frac{1}{P_{c,i}\, T_{c,i}}\right)/n_{c,i} - (\mathrm{A_i D_i} \log(S_i + S_{i+1})\left(\sum_{c=1}^{n_{c,i}}\left(1 - e^{-\lambda_{c,i} P_{c,i}\, T_{c,i}}\right)\left(1 - e^{-P_{c,i}\, T_{c,i} / \beta_i}\right)\right)/\left(\sum_{c=1}^{n_{c,i}} P_{c,i}\, T_{c,i}\right)/n_{c,i}\right)/S_i\right) + 1/\left(\left(\sum_{c=1}^{n_{c,1}} \frac{1}{P_{c,1} T_{c,1}}\right)/n_{c,1} - (\mathrm{A_1 D_1} log(n_e + S_1 + S_2))\left(\sum_{c=1}^{n_{c,1}}\left(1 - e^{-\lambda_{c,1} P_{c,1}\, T_{c,1}}\right)\left(1 - e^{-P_{c,1}\, T_{c,1} / \beta_1}\right)\right)/\left(\sum_{c=1}^{n_{c,1}} P_{c,1} T_{c,1}\right)/n_{c,1}\right)/S_1\right) \quad (5.10)$$

r_e is calculated assuming that the different implemented reasoning models at NMemAgents have 100% prediction accuracy (i.e., A =1).

- We have the following assumptions in our analytical model:
- Each NetMem system level (i) consists of homogeneous NMemAgents (same defined criteria and storage time period for each D/S class in each NMemAgent within same level (i)).
- Message service rate μ_i per NetMem system level (i) is greater than message arrival rate λ_i/S_i (not have saturated NetMem system and to have load balancing among NMemAgents within N levels)
- Based on the used semantics reasoning model (i.e., the value of class existence probability$\boldsymbol{P_{c,i}}$), the response time of the NetMem system can be improved.
- We assume that all classes within NMemAgents located at the same NetMem system level have same $\boldsymbol{P_{c,i}}$

NetMem Efficiency

NetMem efficiency is defined as its ability to retrieve required information or concept classes in a time comparable with the estimated response time of no information misses. We assume that the average NetMem system throughput (T_{hp}) will equal semantics message size (m_s) transmitted over NetMem system levels over the average system response time (r).

We have an assumption that the average NetMem system throughput (T_{hp}) will equal semantics message size (m_s) transmitted over NetMem system levels over the average system response time (r).

- $T_{hp} = m_s / r$ (5.11)

  (Ignoring the latency due to communication channels between NetMem system and network entities)

- System efficiency (performance) = ***NetMem_eff*** = estimated throughput / actual throughput = $(m_s / r_e) / (m_s / r_a)$ = actual response time / estimated response time

  NetMem_eff $= r_a / r_e$ (5.12)
- NetMem system information loss can be calculated as 1- ***NetMem_eff*** where ***NetMem_eff*** express the ability of NetMem to provide required information.
- The estimated response time r_e of NetMem system to retrieve required data is calculated as described in the equation (5.10), $r_e = \sum_{i=1}^{N} r_i$ (we call that estimated response time that we assume that all agents in the system, whatever reasoning model is used, will learn and have the required data with different levels of abstraction with 100 % prediction accuracy).
- The actual response time r_a of NetMem system using a specific reasoning model to get required data is calculated using the last equation, however, the accuracy of each semantics reasoning model is included in the equation. For example, NetMem system with LDA-HMM model will result that 0.9875 (this value is captured based on simulation results with Snort where "LDA→ 0.95, HMM→0.7968") of packets or messages transferred in the system are related to the required data.
- The actual response time r_a of NetMem system can be calculated as follows:

$$
\begin{aligned}
&r_a = \sum_{i=1}^{N} r_i = \\
&1/\left(\left(\sum_{c=1}^{n_{c,N}} \frac{1}{P_{c,N}\, T_{c,N}}\right)/n_{c,N}\right) - \\
&(A_N D_N \log(S_N)(\sum_{c=1}^{n_{c,N}} \left(1 - e^{-\lambda_{c,N} P_{c,N}\, T_{c,N}}\right)(1 - e^{-P_{c,N}\, T_{c,N}/\beta_N}))/(\sum_{c=1}^{n_{c,N}} P_{c,N}\, T_{c,N})/ \\
&n_{c,N})/S_N) + \sum_{i=2}^{N-1}(1/\left(\left(\sum_{c=1}^{n_{c,i}} \frac{1}{P_{c,i}\, T_{c,i}}\right)/n_{c,i} - (A_i D_i \log(S_i + S_{i+1})(\sum_{c=1}^{n_{c,i}} \left(1 - \right.\right. \\
&\left.\left. e^{-\lambda_{c,i} P_{c,i}\, T_{c,i}}\right)(1 - e^{-P_{c,i}\, T_{c,i}/\beta_i}))/(\sum_{c=1}^{n_{c,i}} P_{c,i}\, T_{c,i})/n_{c,i})/S_i)) + 1/\left(\left(\sum_{c=1}^{n_{c,1}} \frac{1}{P_{c,1}\, T_{c,1}}\right)/\right. \\
&\left. n_{c,1} - (A_1 D_1 \log(n_e + S_1 + S_2))(\sum_{c=1}^{n_{c,1}} (1 - e^{-\lambda_{c,1} P_{c,1}\, T_{c,1}})(1 - e^{-P_{c,1}\, T_{c,1}/\beta_1}))/\right. \\
&\left.(\sum_{c=1}^{n_{c,1}} P_{c,1}\, T_{c,1})/n_{c,1})/S_1\right)
\end{aligned}
\qquad (5.13)
$$

5.3.3 Data Classes Abstraction Process

We propose an analytical model for the abstraction processes within the NetMem system. We adopt that analytical model to predict NetMem system processing time and utilization (utl) in the case of building semantics or concept classes. We assume that the total processing time (***p_t***) taken by a NMemAgent to reason about ***n*** concept classes equals ***p_t***. From the queuing theory [120] and inspired by the work at [121] we propose that the expected processing time ***p_t*** of an agent in the NetMem system level (i) (at system service rate $\boldsymbol{\mu}$ > data arrival rate $\frac{\boldsymbol{\lambda}}{\boldsymbol{S}}$) equals $\boldsymbol{p_t = \frac{1}{\mu - \frac{\lambda}{N}}}$ (not to have saturated system with unbounded ***p_t***). This assumption comes with the queuing theory of a system of with M/M/1 queue that the expected response time of an agent within a system equals $\frac{\mathbf{1}}{\boldsymbol{\mu - \lambda}}$. Assuming a

multi-agent system of N agent, then the expected arrival rate to an agent will be $\frac{\boldsymbol{\lambda}}{\boldsymbol{N}}$ (distributed load).

NetMem Processing Time

- The processing time taken by a NMemAgent at level (i) to reason about $\boldsymbol{f_{c,i}\, n_{c,i}}$ concept classes equals $\boldsymbol{pt_i}$ where $\boldsymbol{f_{c,i}}$ is the extraction factor according to the used reasoning model at level (i) and ($0 < f_{c,i} \leq 1$).
- For parallel processing in the one agent's level and for N levels and assuming that agents in the different levels send updates continually for agents in the upper levels, the total processing time to reason about $\boldsymbol{f_{c,N}\, n_{c,N}}$ classes equals $\sum_{\boldsymbol{i=1}}^{\boldsymbol{N}} \boldsymbol{pt_i}$
- Average NetMem system processing time = $\sum_{\boldsymbol{i=1}}^{\boldsymbol{N}} \boldsymbol{pt_i / f_{c,N}\, n_{c,N}}$
  - Using the queuing theory, the processing time pt_i of an agent at NetMem system level (i) is calculated assuming that: 1) level (i) service rate $\mu_i >$ data arrival rate $\frac{\lambda_i}{S_i}$ and 2) the expected generated DU will be distributed over $S_i$agents at level (i) in order to not having saturated NetMem system with unbounded pt_i.
  - $pt_i = \frac{1}{\mu_i - \frac{\lambda_i}{S_i}}$ (time unit) where $\frac{\lambda_i}{S_i}$: DU arrival rate to an agent at level (i) and μ_i equals $\left(\sum_{c=1}^{f_{c,i}\, n_{c,i}} \frac{1}{P_{c,i}\, T_{c,i}}\right) / f_{c,i}\, n_{c,i}$ where:
    - $\boldsymbol{\mu_i}$: the service rate at an agent in NetMem system level (i) (we consider a parallel processing in all agents at level (i), So, $\boldsymbol{\mu_i}$ refers to the service rate at level (i))
    - $\boldsymbol{\lambda_i}$: data generation rate from all possible data sources and related to various classes to agents at level (i)
    - $\boldsymbol{S_i}$: number of agents at level (i)
    - $\frac{\boldsymbol{\lambda_i}}{\boldsymbol{S_i}}$: data arrival rate to an agent at level (i) assuming that the expected generated data will be distributed over $\boldsymbol{S_i}$agents at level (i)

  Note: $\boldsymbol{pt_i}$ represents the processing time of level (i) in case of having parallel processing in all agents at level (i)
- Level (i) service rate $\boldsymbol{\mu_i}$ per agent for providing the service (i.e., extracting $\boldsymbol{f_{c,i}\, n_{c,i}}$ concept classes) where $n_{c,i}$ is the targeted number of classes and $f_{c,i}$ is the extraction factor according to the used reasoning model ($0 < \boldsymbol{f_{c,i}} \leq 1$)
  - $\mu_i = \left(\sum_{c=1}^{f_{c,i}\, n_{c,i}} \frac{1}{P_{c,i}\, T_{c,i}}\right) / f_{c,i}\, n_{c,i}$ (5.14)
    - $\left(\sum_{c=1}^{f_{c,i}\, n_{c,i}} \frac{1}{P_{c,i}\, T_{c,i}}\right)$: the total service rate considering the storage duration $\boldsymbol{T_{c,i}}$ of each concept class and the probability $\boldsymbol{P_{c,i}}$ of class existence at level (i) agents.

- $f_{c,i}\, n_{c,i}$: represents the actual number of concept classes that can be identified at level i based on the prediction accuracy of the used reasoning model in NMemAgents at level i.

▪ Data arrival rate to an agent in NetMem system level (i) = $\frac{\boldsymbol{\lambda_i}}{\boldsymbol{S_i}}$ (arrival rate /class)

– Data generation rate from possible sources to agents at level (i) equals

$$R_i K_i log(S_i + S_{i-1}) \left(\sum_{c=1}^{f_{c,i}\, n_{c,i}} (1 - e^{-\lambda_{c,i} P_{c,i}\, T_{c,i}})(1 - e^{-P_{c,i}\, T_{c,i} / \beta_i})\right) / \left(\sum_{c=1}^{f_{c,i}\, n_{c,i}} P_{c,i}\, T_{c,i}\right) / f_{c,i} n_{c,i} \quad (5.15)$$

where :

- $R_i K_i log(S_i + S_{i-1})$: represents the number of data units which are related to various classes and sent by possible data sources in the same level and the level underneath.
  - $\boldsymbol{R_i}$: redundancy factor that represents the replica of data units of targeted classes that can be generated and sent to level (i) = $\left(\sum_{c=1}^{f_{c,i}\, n_{c,i}} R_{c,i}\right) / f_{c,i} n_{c,i}$
  - $\boldsymbol{K_i}$: average number of generated data units from data sources per class tp agents at level (i)
  - $log(S_i + S_{i-1})$: number of data sources for an agent at level (i) according to have a connected random graph with high probability
    - $\sum_{c=1}^{f_{c,i}\, n_{c,i}} (1 - e^{-\lambda_{c,i} P_{c,i}\, T_{c,i}})(1 - e^{-P_{c,i}\, T_{c,i} / \beta_i})$: is the total probability of having non-zero generated data units related to the targeted $\boldsymbol{f_{c,i} n_{c,i}}$ classes within the storage duration of each concept class at level (i) agents.
      - $(1 - e^{-\lambda_{c,i} P_{c,i}\, T_{c,i}})$: the probability of having non-zero generated data units related to a certain class and sent to level (i) agents where $\boldsymbol{\lambda_{c,i}}$ represents the arrival rate of data units related to a certain class through the storage duration of that class and $\boldsymbol{\lambda_{c,i} P_{c,i}\, T_{c,i}}$ is the expected number of arrived class data units through the storage duration of that class
      - $(1 - e^{-P_{c,i}\, T_{c,i} / \beta_i})$: the probability that a data unit related to an update in a certain concept class will be sent through the storage duration of that class at level (i) agents and β_i is the average storage duration at level (i) equals $\left(\sum_{c=1}^{f_{c,i}\, n_{c,i}} P_{c,i}\, T_{c,i}\right) / f_{c,i} n_{c,i}$

- $(1 - e^{-\lambda_{c,i} P_{c,i} T_{c,i}})$ and $(1 - e^{-P_{c,i} T_{c,i} / \beta_i})$ are considered as probabilities of two independent events
- $\left(\sum_{c=1}^{f_{c,i}} n_{c,i} P_{c,i} T_{c,i}\right) / f_{c,i} n_{c,i}$: it represents the average service time per class when it is expected to receive generated data units related to various concept classes.

- Data arrival rate to an agent at level (i)=

$$(R_i K_i log(S_i + S_{i-1}) \ (\sum_{c=1}^{f_{c,i} n_{c,i}} (1 - e^{-\lambda_{c,i} P_{c,i} T_{c,i}})(1 - e^{-P_{c,i} T_{c,i} / \beta_i}) \ ) / \left(\sum_{c=1}^{f_{c,i} n_{c,i}} P_{c,i} T_{c,i}\right) / f_{c,i} n_{c,i}) / S_i \quad (5.16)$$

Assuming that the generated data units are sent and arrived to level (i) and distributed over S_i agents

NetMem processing time to reason about $n_{c,N}$ concept classes in N NetMem levels is computed as follows:

$$pt = \sum_{i=1}^{N} pt_i =$$
$$\sum_{i=2}^{N} (1/(\left(\sum_{c=1}^{f_{c,i} n_{c,i}} \frac{1}{P_{c,i} T_{c,i}}\right) / f_{c,i} n_{c,i} - (R_i K_i log(S_i + S_{i-1}) \ (\sum_{c=1}^{f_{c,i} n_{c,i}} (1 -$$
$$e^{-\lambda_{c,i} P_{c,i} T_{c,i}})(1 - e^{-P_{c,i} T_{c,i} / \beta_i})) / \left(\sum_{c=1}^{f_{c,i} n_{c,i}} P_{c,i} T_{c,i}\right) / f_{c,i} n_{c,i}) / S_i + 1 / \quad (5.17)$$
$$(\left(\sum_{c=1}^{f_{c,1} n_{c,1}} \frac{1}{P_{c,1} T_{c,1}}\right) / f_{c,1} n_{c,1} - (R_1 K_1 log(n_e + S_1) \ (\sum_{c=1}^{f_{c,1} n_{c,1}} (1 -$$
$$e^{-\lambda_{c,1} P_{c,1} T_{c,1}})(1 - e^{-P_{c,1} T_{c,1} / \beta_1})) / \left(\sum_{c=1}^{f_{c,1} n_{c,1}} P_{c,1} T_{c,1}\right) / f_{c,1} n_{c,1}) / S_1$$

- The estimated maximum processing time (est_pt)
  - (extraction factor) $\boldsymbol{f_{c,i}} = 1$ for all levels (i.e., each level i will have the expected number of concept classes)
  - (existing probability) $P_{c,i} = 1$ for all agents at all levels regardless of the used reasoning model
  - (Redundancy) $R = 1$ for all traffic from all levels (i.e., redundancy decreasing factor)
- The actual processing time (act_pt)
  - $\boldsymbol{f_{c,i}} \leq 1$ for all levels (i.e., each level i might have the expected number of concept classes) → this depends on the used reasoning model (LDA-HMM, HMM, LDA) and the usage of LSH or not
  - (existing probability) $P_{c,i} \leq 1$ for all agents in all levels relied on the used reasoning model
  - (Redundancy) $R < 1$ for all traffic form all levels (i.e., redundancy decreasing factor)

NetMem Utilization

We assume that the average processing time for extracting a concept class at the N-level equals total NetMem processing time/number of extracted classes $(n_{c,N})$ at the N^{th} NetMem level (i.e., the average processing time equals $pt/\ n_{c,N}$).

- utl = the ratio of the average processing time NetMem system in use to extract a concept class at the N-level to max. average processing time that NetMem system could be in use to extract a concept class at the N-level
- NetMem system efficiency (eff) can be calculated in terms of NetMem utilization. In other words, eff is computed as the difference between the max. and actual average processing time over the max. avg. processing time (1-utilization).
- eff is defined as the ability of the system to reason about targeted number of classes in shorter actual processing time compared with the maximum processing time.

5.3.4 Optimization Problem for NetMem Configuration

5.3.4.1 Problem Formulation

In NetMem, the shared distributed multi-agent multi-level intelligent system, there is a group of algorithms (three algorithms: HMM, LDA, LDA-HMM) that can be implemented over $\boldsymbol{S}$ agents in the different $\boldsymbol{N}$ levels and used for semantics reasoning. So, we formulate an optimization problem for seeking the best distribution of those algorithms over the $\boldsymbol{S}$ agents in the $\boldsymbol{N}$ levels in order to maximize system efficiency at meeting queries of information retrieval. Our assumption is that all $\boldsymbol{S_i}$ agents in the same i^{th} level ($1 \leq i \leq N$) will utilize the same algorithm for semantics reasoning. Some related work, see for example [122], targeted the derivation of efficient data retrieval query processing strategies.

Allocating semantics reasoning algorithms over agents can be modeled as a task allocation or assignment problem (considered as one of the combinatorial problems [123]) in multi-agent systems. We consider that search for semantics reasoning algorithms among various ones and list them over agents in different $\boldsymbol{N}$ levels as a task assignment process. Static and dynamic task allocation and assignment problems have many classes that relate to NP-hard class problem [124-126]. Due to lack of non-polynomial algorithms that can solve NP-hard problems to optimality [127], we utilize Genetic Algorithm (GA) [78] as a heuristic algorithm that can provide optimal or sub-optimal solutions for our problem in polynomial time. GA is one of the evolutionary algorithms that is used to search for the solutions that enhance the data semantics throughput of the NetMem system depending on the principal of "Survival of the fittest solution". Some related optimization problems (e.g., [122]) were formulated for improving the information query processing in distributed database systems. The authors utilized GA to get near-optimal solutions in polynomial time.

a) Fitness Function

The fitness function is to maximize NetMem system efficiency (***NetMem_Eff***) in order to find and retrieve required concept classes with high probability. The proposed fitness function is subjected to some constraints, described shortly.

Max (***NetMem_Eff***) = $\boldsymbol{r_a / r_e = \sum_{i=1}^{N} r_{a,i} / \sum_{i=1}^{N} r_{e,i}}$ for $\forall N$ NetMem levels (5.18)

where $\boldsymbol{r_e}$ and $\boldsymbol{r_a}$ are defined in (5.10) and (5.13) respectively.

Our problem depends on the following parameters and constraints:

b) Constraints

1) $\boldsymbol{0 \leq NetMem\_eff \leq 1}$ (the range of the objective function value)
   - » ***Where*** $\boldsymbol{0 \leq \overline{r_{a,i}} \leq \overline{r_{e,i}}}$
2) $\boldsymbol{0 \leq p_R \leq 1}$ (the prediction accuracy range)
3) $\boldsymbol{T_c > 0}$ (non-negative and non-zero values for storage time of data semantics or concept classes)
4) $\boldsymbol{S_i > 0}$ and $\boldsymbol{S_i > S_j}$ ***where*** $\boldsymbol{j > i}$ (non-negative and non-zero values for number of NetMem agents).
5) $\boldsymbol{n_{c,i} > 0}$ and $\boldsymbol{n_{c,i} < n_{c,j}}$ ***where*** $\boldsymbol{j > i}$ (non-negative and non-zero values for number of defined data semantics classes).
6) $\boldsymbol{n_e > 0}$ (non-negative and non-zero values for number of network entities).

c) Optimization Algorithm

We applied GA as a heuristic approach for getting optimal or sub-optimal solutions for the NConOP problem.

– Chromosome (C) Design:

Each chromosome represents a possible solution where it contains a group of genes (i.e., recommended reasoning models in each level). The length of each chromosome (i.e., number of genes) equals the number of NetMem levels. Possible reasoning models are encoded as a vector of real numbers where each number refers to reasoning model prediction accuracy. For instance, a chromosome C for 6-level NetMem system and p_R:{p_{HMM} =0.95, p_{LDA} =0.92, $p_{LDA\text{-}HMM}$=0.9} might be {p_{R1}=0.9,p_{R2}=0.92,p_{R3}=0.95,p_{R4}=0.95,p_{R5} =0.95,p_{R6}=0.95} where p_{Ri} is the prediction accuracy of the recommended reasoning model at level i.

– GA operations:

Figure 5.2 illustrates the pseudo code of the implemented GA. We adapted the GA Java code presented at [128] to suit solving the NConOP problem.

5.3.4.2 The Genetic Algorithm (GA)

The GA is one of the evolutionary optimization algorithms and it is based over a biological metaphor [117]. The recommended solutions are chosen depending on their fitness with the fitness (objective) function. Initial population (solutions) is generated random with size $\boldsymbol{p_{size}}$. Each solution is represented by a chromosome of a fixed length

where each chromosome is a vector of parameters (genes). Each gene indicates to a certain dependent or independent element of the fitness function. The GA depends on three operators, which are the selection, crossover and mutation. The selection operator is related with the selection of the best solutions (chromosomes) due to their fit to the fitness function. The selection is executed due to a defined probability P_s which defines the ratio of parent solutions that will be used for next generations of solutions. The crossover operator tells that a pair of the best candidates' solutions are chosen and mated due to a probability P_c. The mutation operator represents a new candidates' solution that are reproduced with probability P_m. The GA can be used for unconstrained and constrained optimization problems. The optimization process measures the performance of each chromosome and takes into consideration the constraints of the problem if it is constrained. The GA represents each generation with new offspring of N chromosomes where crossover and mutation operations are done. The GA process is repeated with certain number of generations or iterations. The GA process is terminated due to certain stopping criteria, such as reaching the defined number of iterations or passing the assigned threshold value of the fitness function.

```
Input: fitness function, population size, solution vector length, iterations number, parent usage ratio, mutation percentage, crossover ratio, FitnessValueThreshold
1     for(population size) {
2               generate random solution S ; determining vector of implemented reasoning models at NetMem levels where each vector element represents a semantics reasoning model at a NetMem level
3               store (S)
4     }
5     Int count = 0;
6     while ((count < iterations number)&&(maxFitnessValue < FitnessValueThreshold)) {; stopping criteria
7               while (new population size < (1- parent usage ratio) * population size ) {
8                         select N random solutions  where N < population size
9                         for (N) {  calculate the value of fitness(N) }
10                        get the best two candidate solutions (Sc1,Sc2) with larger fitness values
11                        get new solutions (Sc1_new,Sc2_new) = crossover (Sc1,Sc2)
12                        generate random number R
13                        if(R < mutation percentage) {   mutation(Sc1_new)   } else {   mutation(Sc2_new)    }
14                        if(fitness(Sc1_new) is better than fitness(Sc1)) {store_new (Sc1_new)} else {store_new (Sc1)}
15                        if(fitness(Sc2_new) is better than fitness(Sc2)) {store_new (Sc2_new)} else {store_new(Sc2)}
16              }
17              for (int i =0 ; i < parent usage ratio * population size; i++ ) {
18                        store_new (population size[i])
19              }
20              maxFitnessValue = 0;
21              for(int j=0, j<new population size;j++) {
22                   do{
23                        for(int k=0, k < new population size[j];k++) {generate random prediction accuracy values based on  model k}
24                        calculate fitness(new population size[j])
25                        if(fitness(new population size[j]) > maxFitnessValue)  {maxFitnessValue= fitness(new population size[j]);}
26                   } while (fitness(new population size[j]) does not meet problem constraints)
27              }
28    Count++;
29    }
30    crossover(S1,S2) {
31              generate random number R
32              if(R > crossover ratio) {S1_new = S1, S2_new = S2}  else{S1_new = S2, S2_new = S1}
33    }

34    mutation(S) {
35              S = generate (random number)*S
36    }

37    Fitness (solution vector) {
38              generate random NetMem level ID; this level ID represents the level where required data/semantics exist
39              calculate fitness value (i.e., NetMem efficiency) with respect to the defined constraints
40              for(solution vector length) {get the most frequent semantics reasoning model and the related NetMem level ID}
41    }

Output: optimal NetMem system configuration solution vectors, problem convergence time
```

Figure 5.2 The pseudo code of the implemented genetic algorithm

5.4 NetMem Efficiency and Improved Configuration

We adopt the previously discussed analytical model to calculate response time of the NetMem system in case of data retrieval process. Also, we calculate NetMem processing time in case of data abstraction process. We consider in our study the variation in the implemented semantics reasoning models (HMM, LDA, and LDA-HMM) in

NMemAgents and the operation with full and reduced dimensional data using LSH algorithms. We calculate the NetMem system efficiency at the following two cases:

1. Required data/information is of high detail (in agents at low levels)
2. Required data/information is of less detail (in agents at higher levels).

5.4.1 Results based on NetMem analytical model for data retrieval process

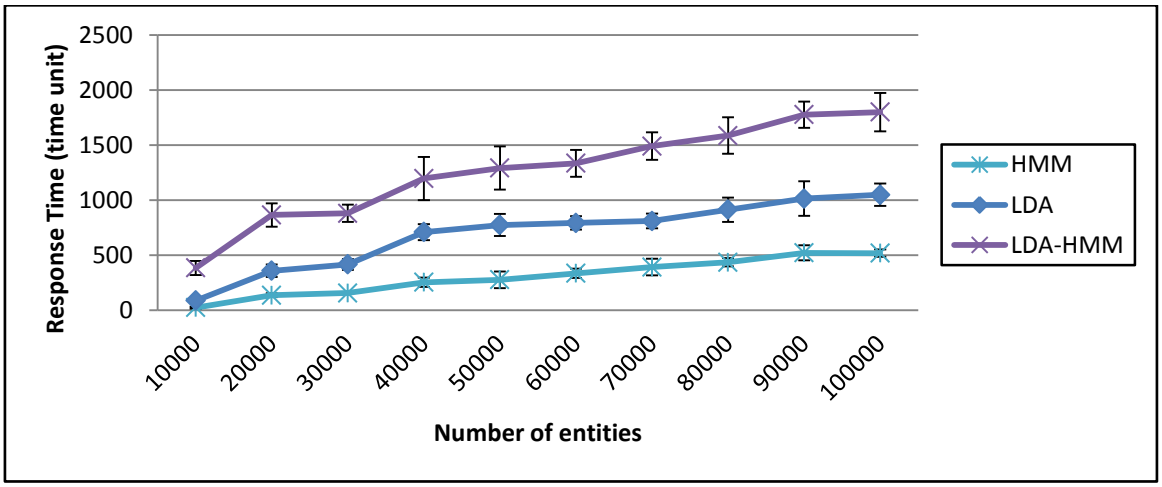
We designed Java classes for implementing the analytical model and its related mathematical equations for calculating the NetMem response time and efficiency. Table 5.1 shows the main parameters and their default values used by the analytical model.

Table 5.1 The parameters and their values used by the analytical model

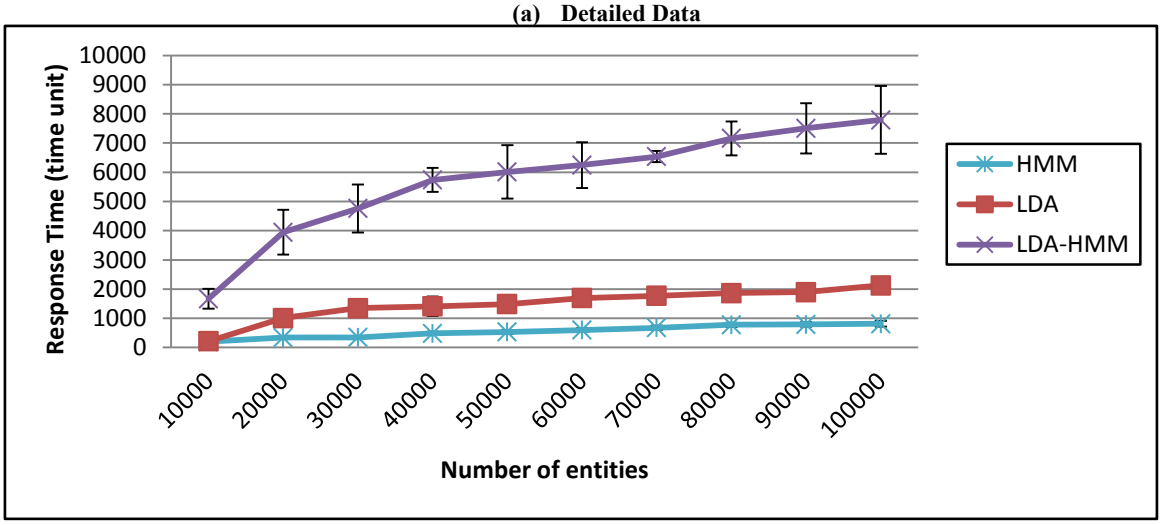
Parameter	Default Value	Range
Number of NetMem system levels (N)	10 levels	3,6,10,20 levels
Number of NMemAgents (S) per NetMem level	Variable (based on the NetMem level)	50-1000 agents
Number of network entities (n_e)	50000 entities	10000 – 100000 entities
Number of identified data/semantics classes per NetMem level (n_c)	Variable (based on the NetMem level)	10 – 100 classes
inter-arrival time of data messages and data/semantics request messages ($1/\lambda$)	Variable	5×10^{-8} – 500 time unit
Percentage of data sources that have available classes (A)	Variable (based on the reasoning model)	0.8 – 1
Probability of data/semantic classes existence (P_c)	Variable (based on abstraction level of classes)	0.1 – 1
Storage duration of concept classes (T_c)	Variable (based on the NetMem level)	4.5×10^{-3} – 36 time unit
Data units per class (D)	one	1 – 100 data unit

5.4.1.1 NetMem Response Time

We tested the NetMem response time at varying number of requesting network entities, number of NMemAgents per each NetMem system level, and the arrival rate of queries for data/semantics classes. Results were obtained with confidence 95%. Figure 5.3 shows NetMem system response time at varying the number of network entities and adopting different semantics reasoning models. The probability of data/semantic classes' existence at using the LDA-HMM based reasoning model is high and also the time complexity of that model is considered. So, NetMem system took much time to retrieve required data especially at abstracted data which are located at higher NetMem system levels. Figure 5.4 shows NetMem response time at increasing the number of NMemAgents in NetMem levels. Increasing the number of agents per levels provided more load balancing and increased the probability of having agents with better service times (i.e., agents have quick service rate or agents have light load).

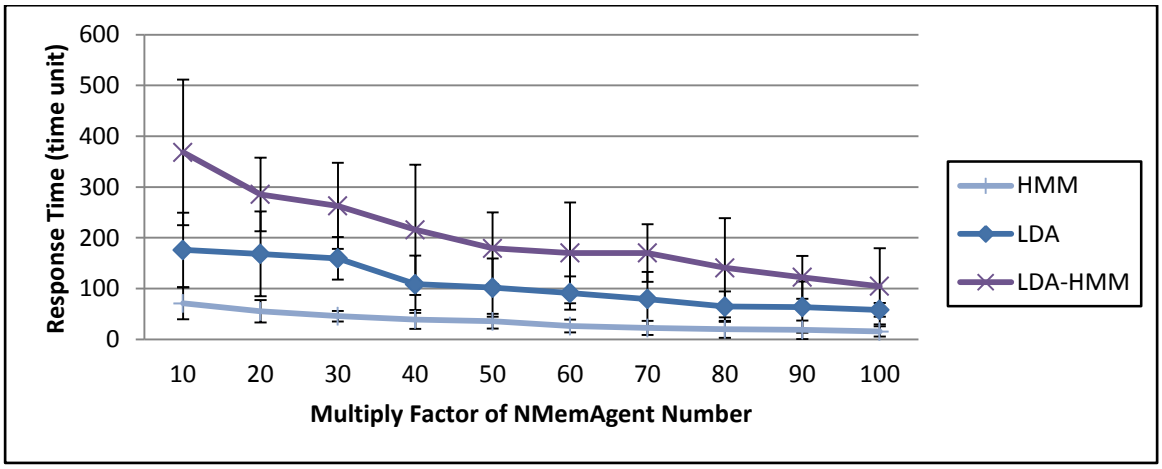


(a) Detailed Data

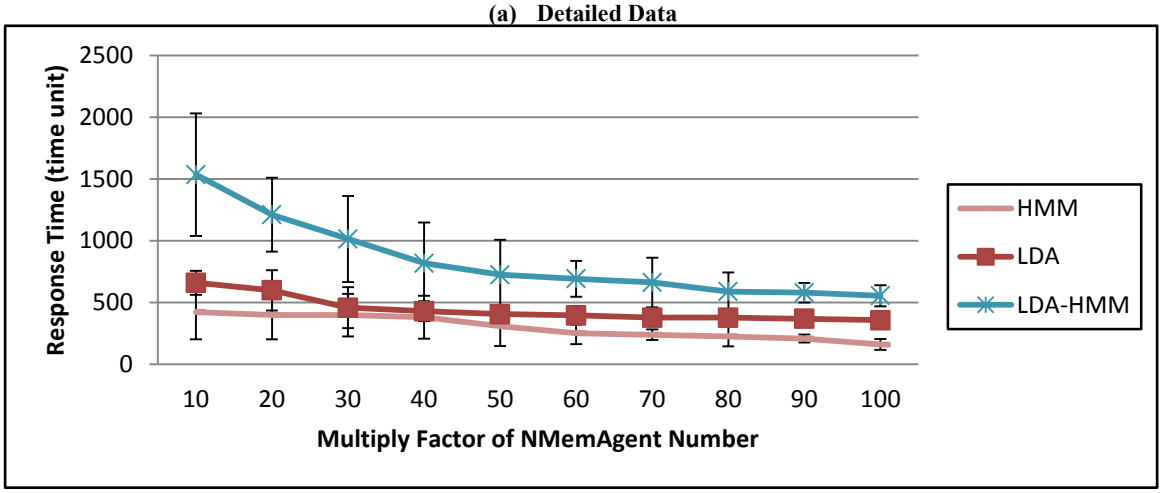


(b) Abstracted Data

Figure 5.3 NetMem response time at varying the number of entities



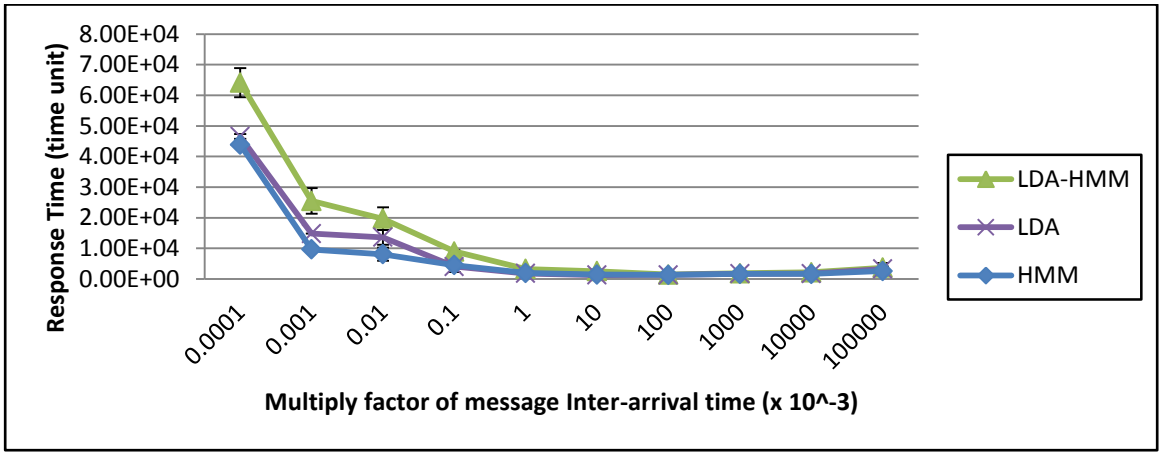
(a) Detailed Data



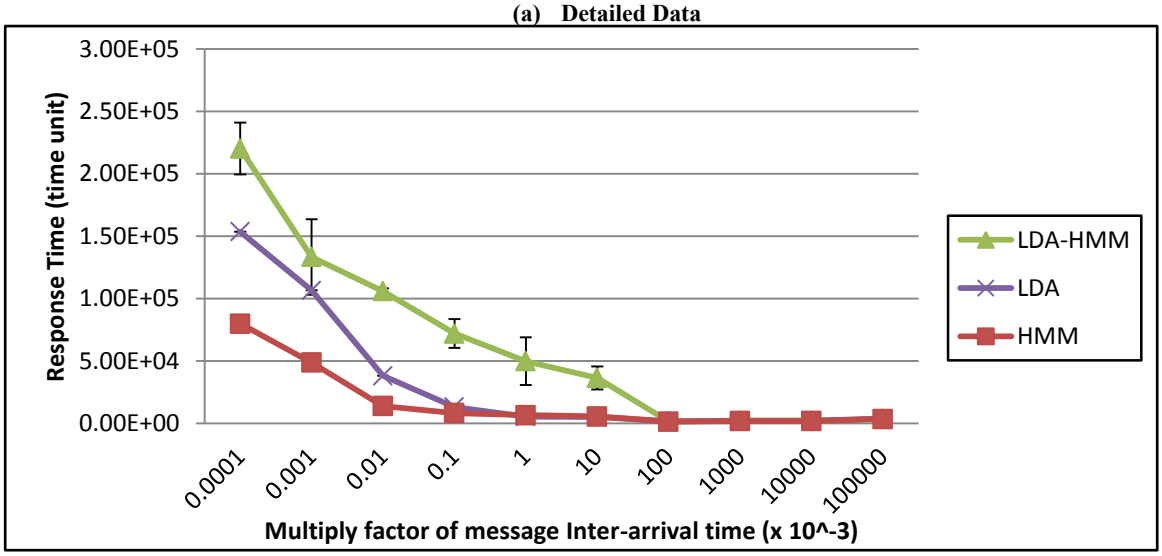
(b) Abstracted Data

Figure 5.4 NetMem response time at varying the number of NMemAgents

Figure 5.5 illustrates the obtained results of the NetMem response time at varying the inter-arrival time of query and data messages to NMemAgents. The longer inter-arrival time of messages is, the shorter response time can be obtained. However, increasing the inter-arrival time might result in some lengthening in the response time since some agents might not have required data and then more messages will be sent to agents in higher levels.



(a) Detailed Data



(b) Abstracted Data

Figure 5.5 NetMem response time at varying the inter-arrival time of query/data messages

Findings and conclusion

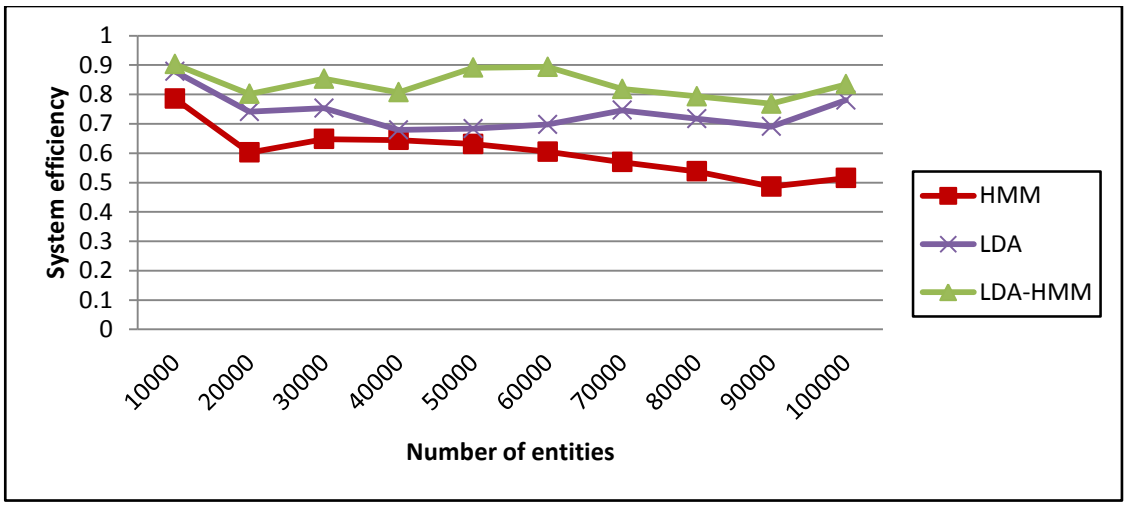
- The response time for retrieving detailed data is better than the one of abstracted data since detailed data are most probable to be found at lower NetMem system levels with shorter-term memory.
- The best model which has minimum overhead was HMM and if required data is found at NetMem using that model, it will enables the system to behave with low response time.
- The response time of NetMem for retrieving abstracted data was the worst at using LDA-HMM since the abstracted data are most probable found at higher NetMem levels and LDA-HMM needs much time to learn and retrieve required abstracted compared with other models.
- The NetMem system with the proposed analytical model for evaluating NetMem

efficiency considered the processing time overhead for different implemented reasoning models besides the probability to find the required data based on data abstraction level.

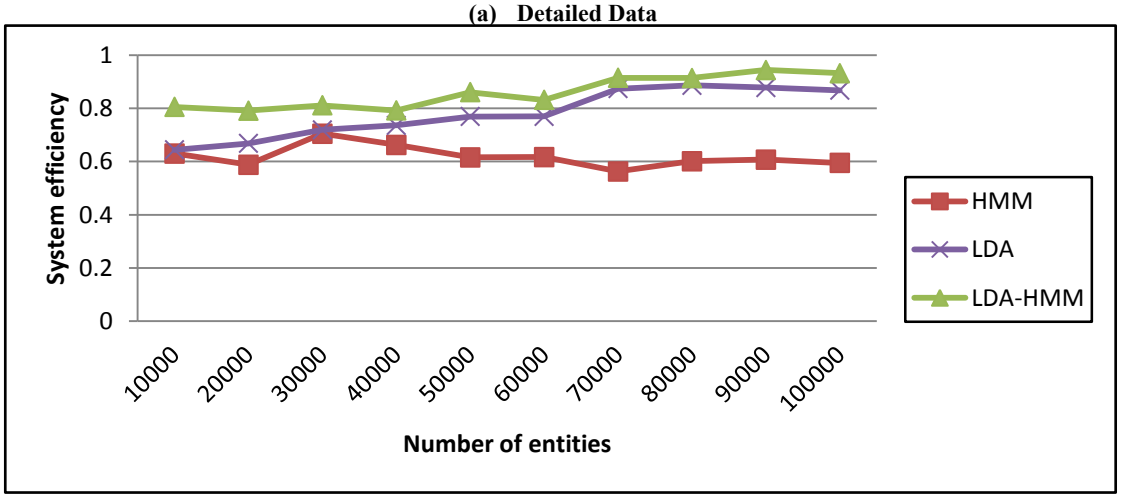
– There was a trade-off between the reliability of the hybrid model to learn required data and time overhead to learn data due to implemented algorithms of both monolithic intelligence techniques (i.e., HMM and LDA).

5.4.1.2 NetMem Efficiency

Figure 5.6 illustrates a comparison between the reasoning models and the achieved NetMem system efficiency at varying the number of network entities. The reasoning models with LDA-HMM and LDA algorithms were better compared with HMM where they achieved high NetMem system efficiency at large number of entities (i.e., system can meet data/semantics of large amounts of entities).



(a) Detailed Data



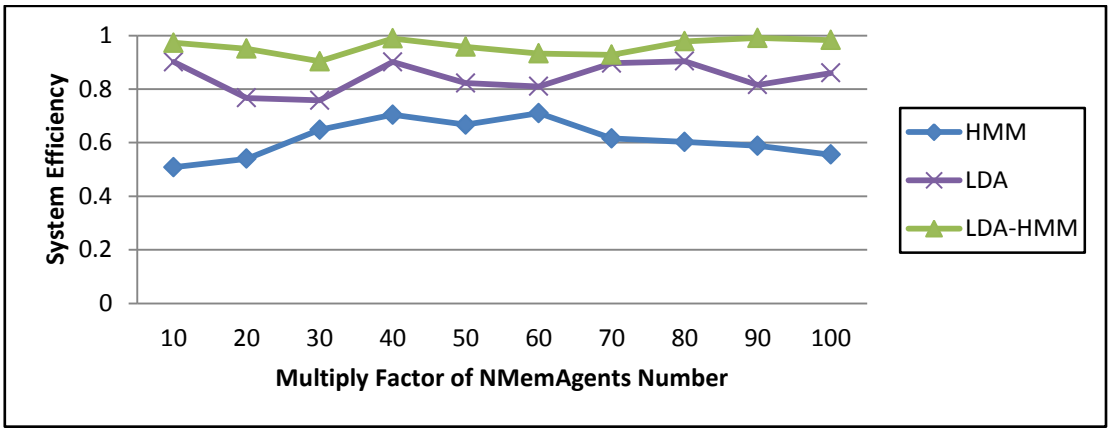
(b) Abstracted Data

Figure 5.6 NetMem system efficiency at varying the number of network entities

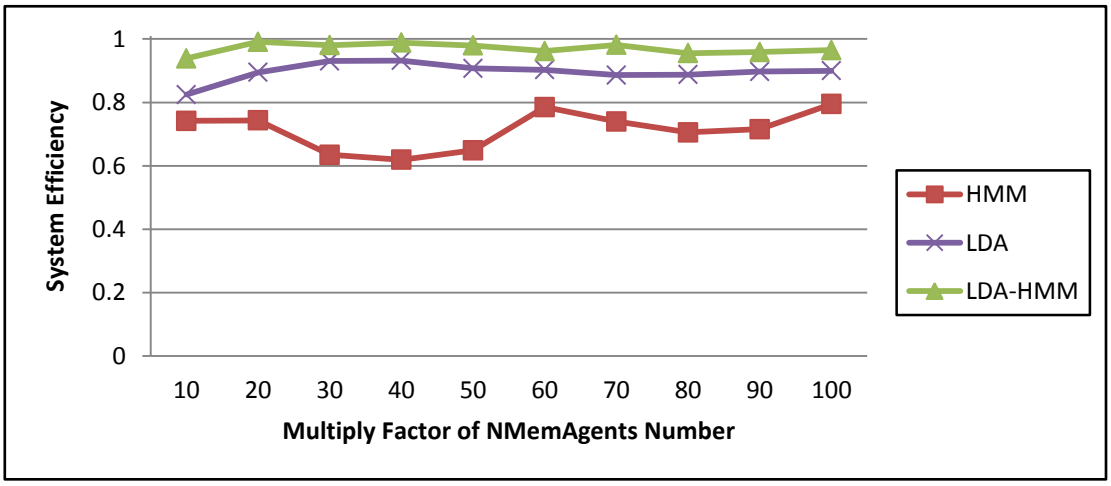
Findings and conclusion

- The hybrid semantics reasoning model with LDA-HMM technique succeeded in achieving the highest NetMem system efficiency compared with reasoning models with HMM or LDA.
- Increasing the number of entities means high probability to have many requests for certain data/semantics and if the system can achieve high efficiency, this means that the system can response with high data/semantics throughput (i.e., many agents have learned required data/semantics).

Figure 5.7 illustrates a comparison between the implemented reasoning models and the achieved NetMem system efficiency at varying the number of NMemAgents.



(a) Detailed Data



(b) Abstracted Data

Figure 5.7 NetMem system efficiency at varying the number of NMemAgents

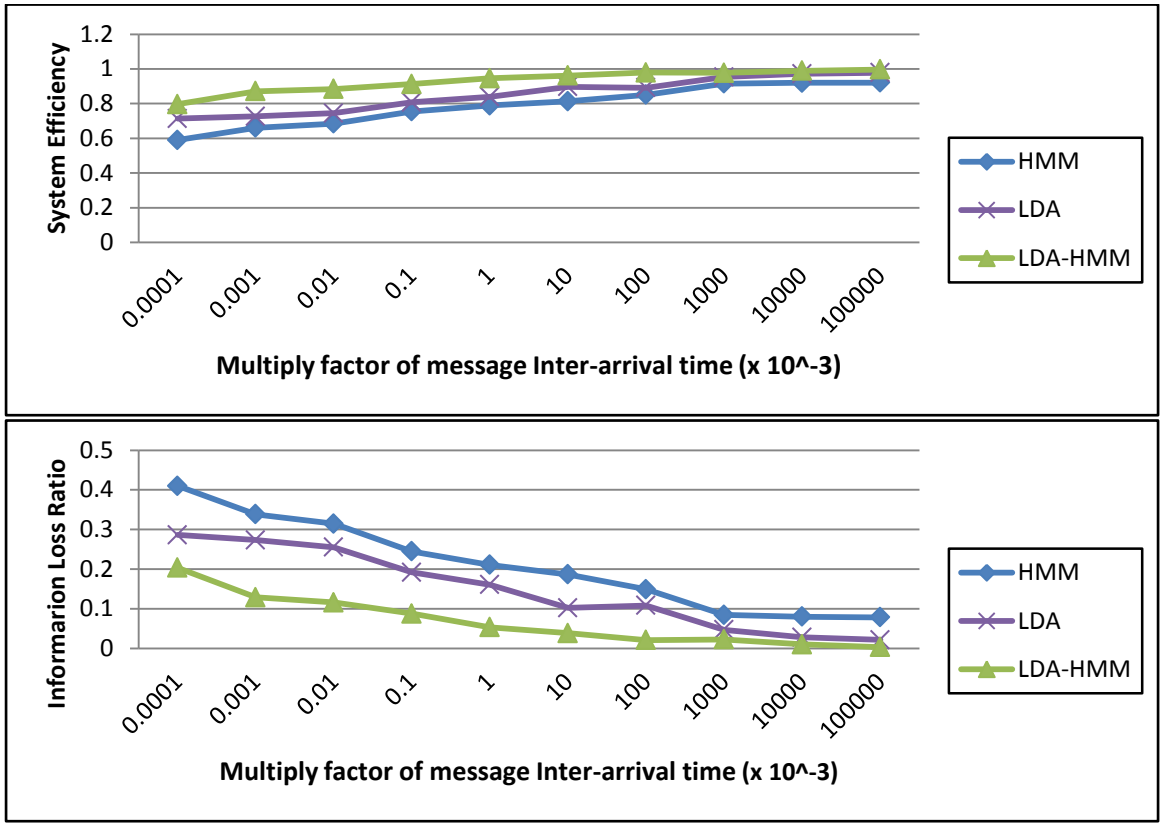
Findings and conclusion

- Increasing number of agents aided in enhancing NetMem system response time at using various implemented reasoning models.
- The response time of NetMem system with the different implemented reasoning models was improved at larger number of NMemAgents compared with the case of operation at 50000 entities with small number of agents.
- The hybrid semantics reasoning model with LDA-HMM technique succeeded in achieving the highest NetMem system efficiency compared with reasoning models with HMM or LDA.

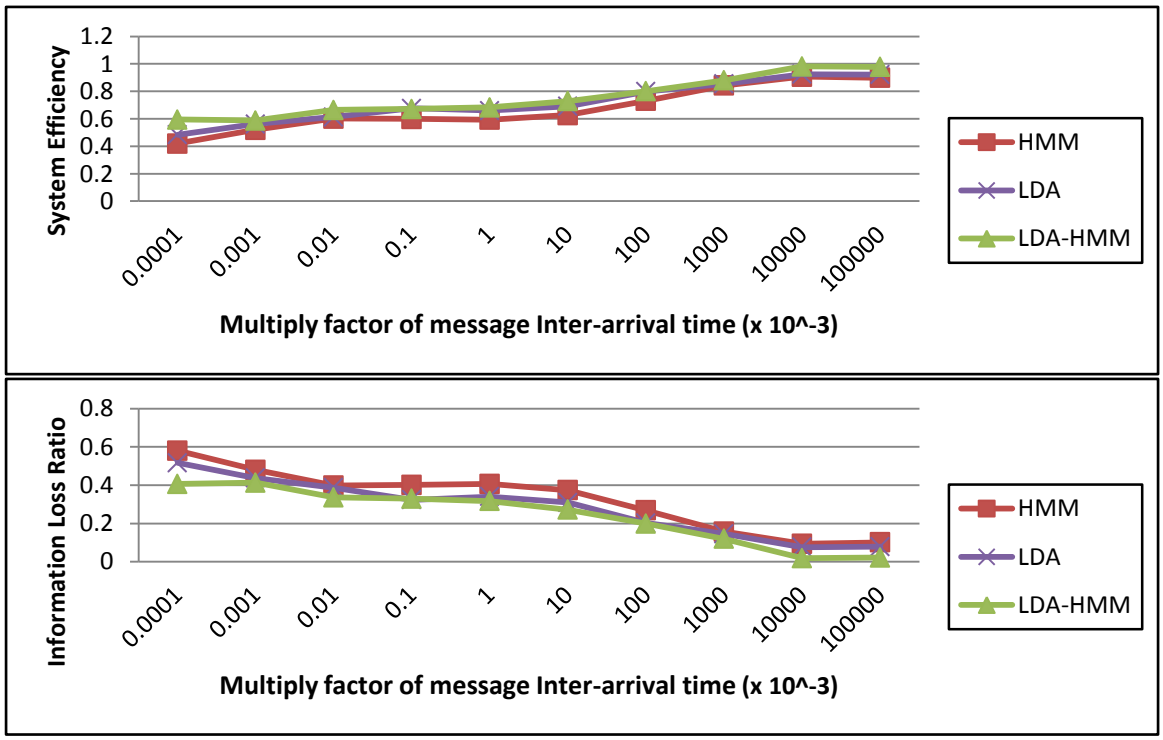
Figure 5.8 illustrates a comparison between the reasoning models and the achieved NetMem system efficiency at varying the inter-arrival time of data messages.

Findings and conclusion

- Shorter NetMem response time at retrieving data/semantics from low NetMem levels.
- The shorter inter-arrival message time is, the longer NetMem response time achieves
- NetMem response time decreases as the inter-arrival time between data messages increases in a certain range
- NetMem response time lengthens as inter-arrival time increases over a specific range
- Data requests take longer response time in case of requesting highly abstracted data (semantics)
- Increasing inter-arrival time of messages (or decreasing rate of arrived messages) helped in improving NetMem system efficiency at using semantics reasoning models (e.g., HMM) with low prediction accuracy
- The hybrid semantics reasoning model with LDA-HMM technique succeeded in achieving the highest NetMem system efficiency compared with reasoning models with HMM or LDA
- There is a capability for choosing a value for the inter-arrival message time which achieves good NetMem system response time, high system efficiency, and low information loss ratio.



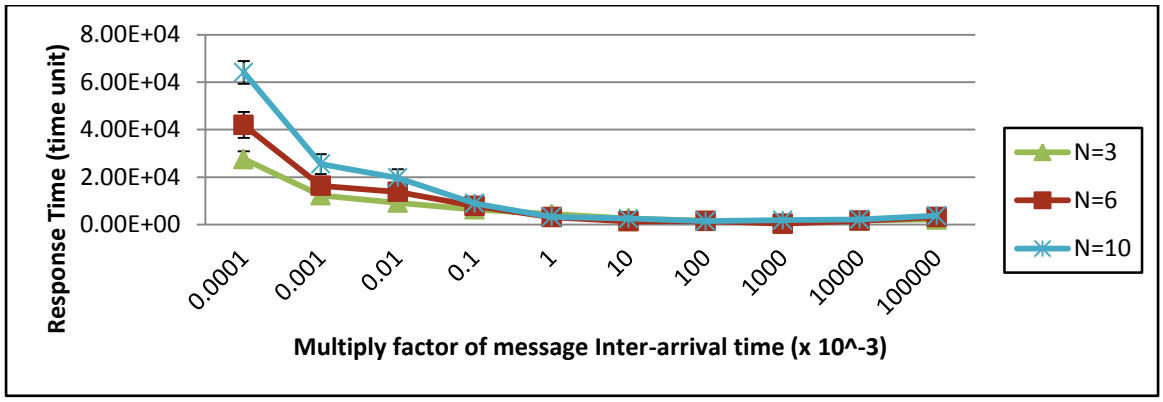
(a) Detailed Data



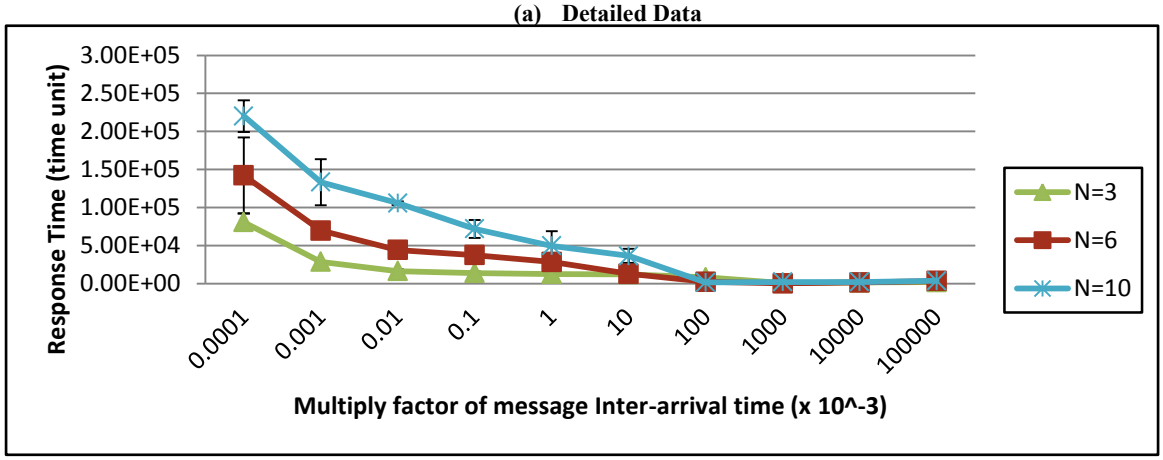
(b) Abstracted Data

Figure 5.8 NetMem system efficiency and information loss ratio at varying the inter-arrival time of query messages

Figure 5.9 illustrates the NetMem system response time at varying the inter-arrival message rate at different number of NetMem levels (three, six and ten (default value)).



(a) Detailed Data



(b) Abstracted Data

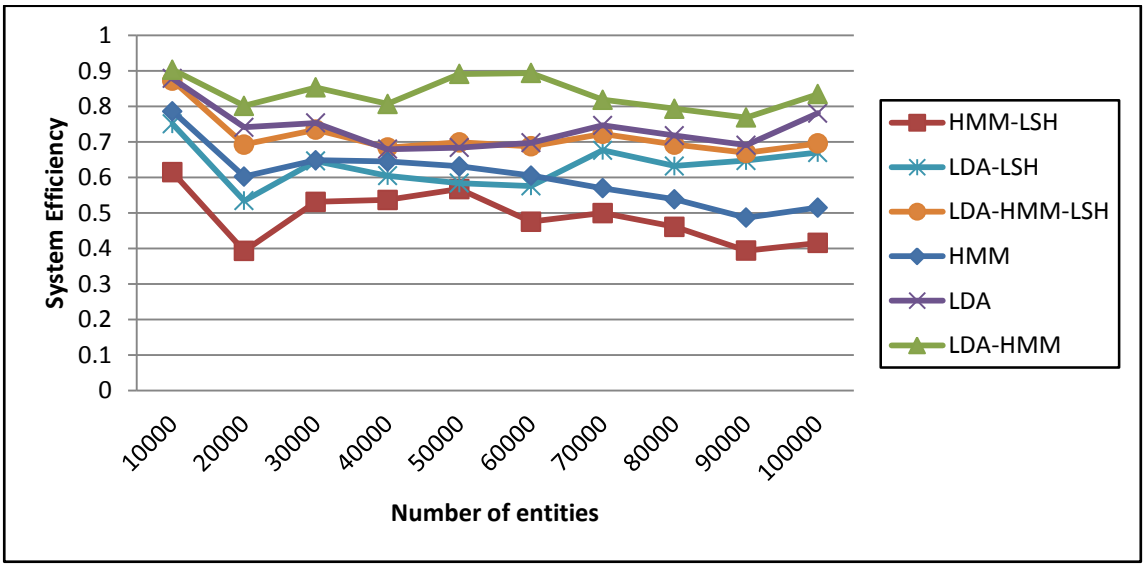
Figure 5.9 NetMem system response time at varying the inter-arrival message rate w.r.t different NetMem level number

Findings

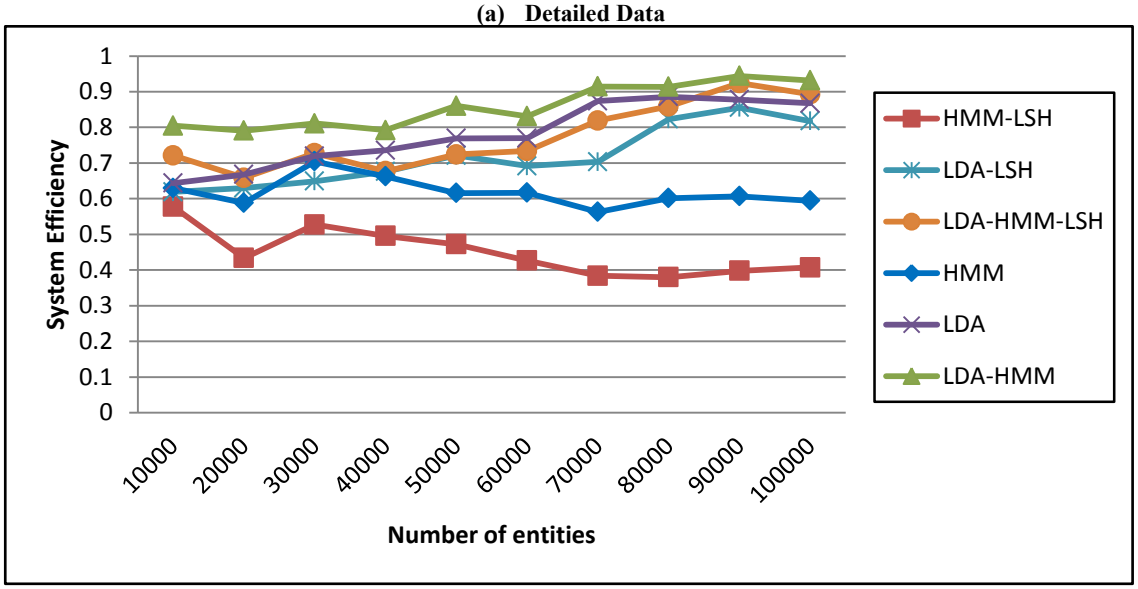
- NetMem response time increases at higher data arrival rate and at large number of NetMem levels
- NetMem response time approaches same values w.r.t. different NetMem levels at lower data arrival rate whether retrieving detailed or abstracted data.

NetMem Efficiency with using LSH Algorithm

We calculated the NetMem system efficiency at using LSH algorithm. In this case, the prediction accuracy of reasoning algorithms is affected since the models learn patterns of reduced dimensional data. We used accuracy values for reasoning models based on results obtained from the simulation studies. We got the actual response time of NetMem based on the new assigned prediction accuracy for the reasoning models. Then, we got the efficiency using the already calculated values of the estimated response time. We made a comparison between NetMem efficiency with/without adopting LSH. Figure 5.10, Figure 5.11 and Figure 5.12 show the obtained results of NetMem system



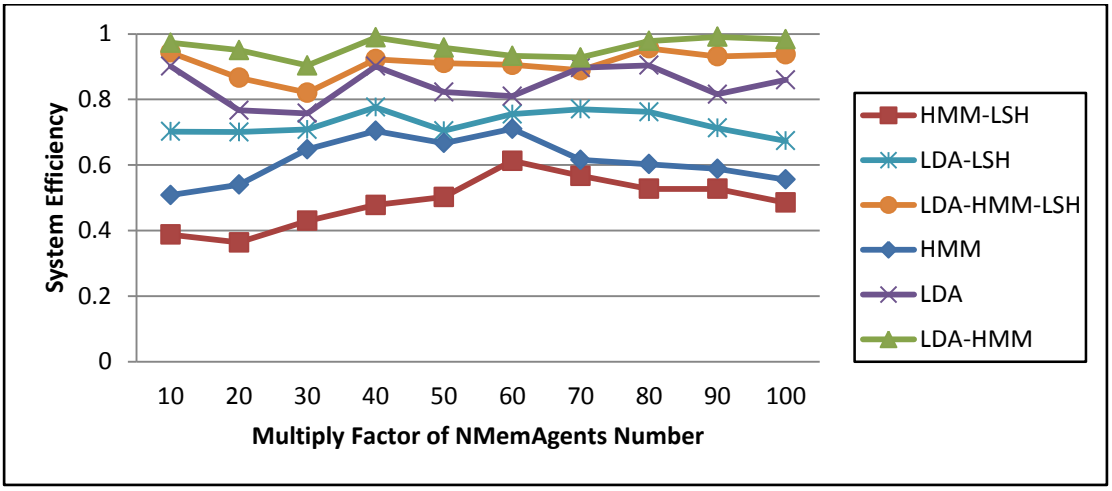
(a) Detailed Data



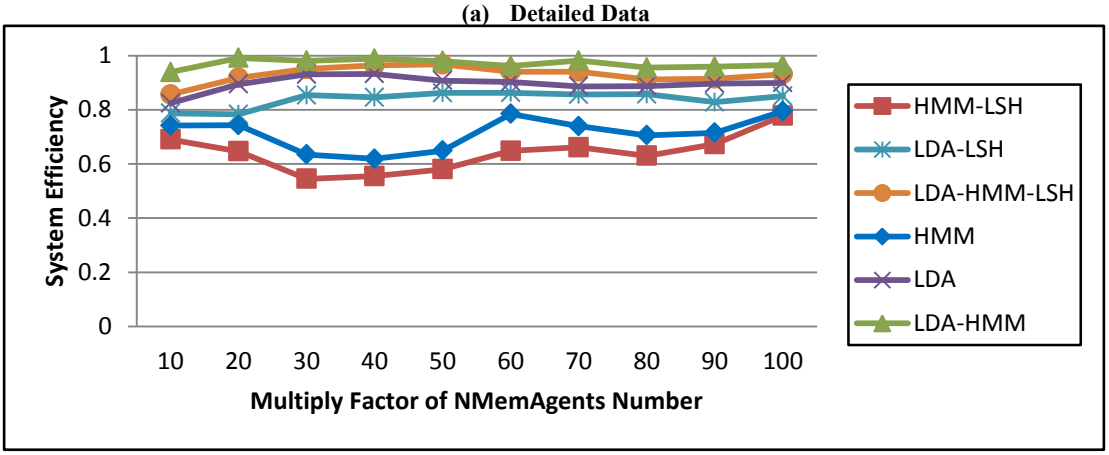
(b) Abstracted Data

Figure 5.10 NetMem system efficiency with/without LSH at varying the number of network entities

efficiency at using various semantics reasoning models with/without adopting LSH algorithms for retrieving detailed or abstracted data.



(a) Detailed Data

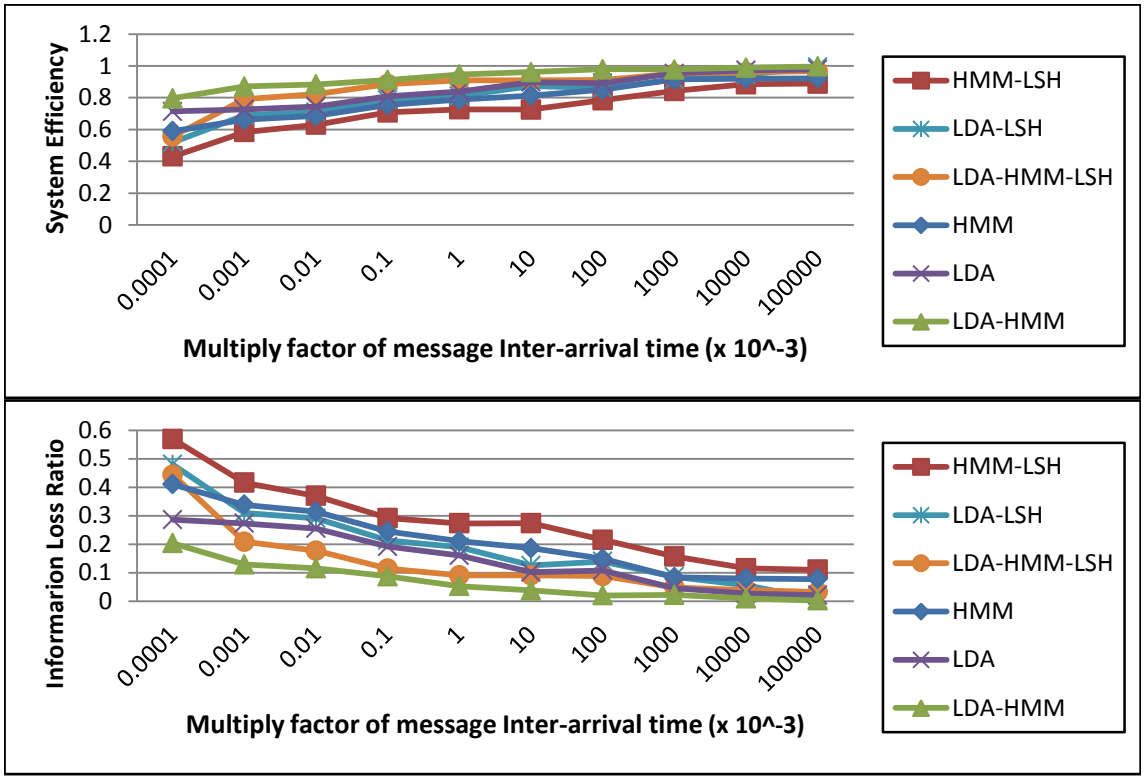


(b) Abstracted Data

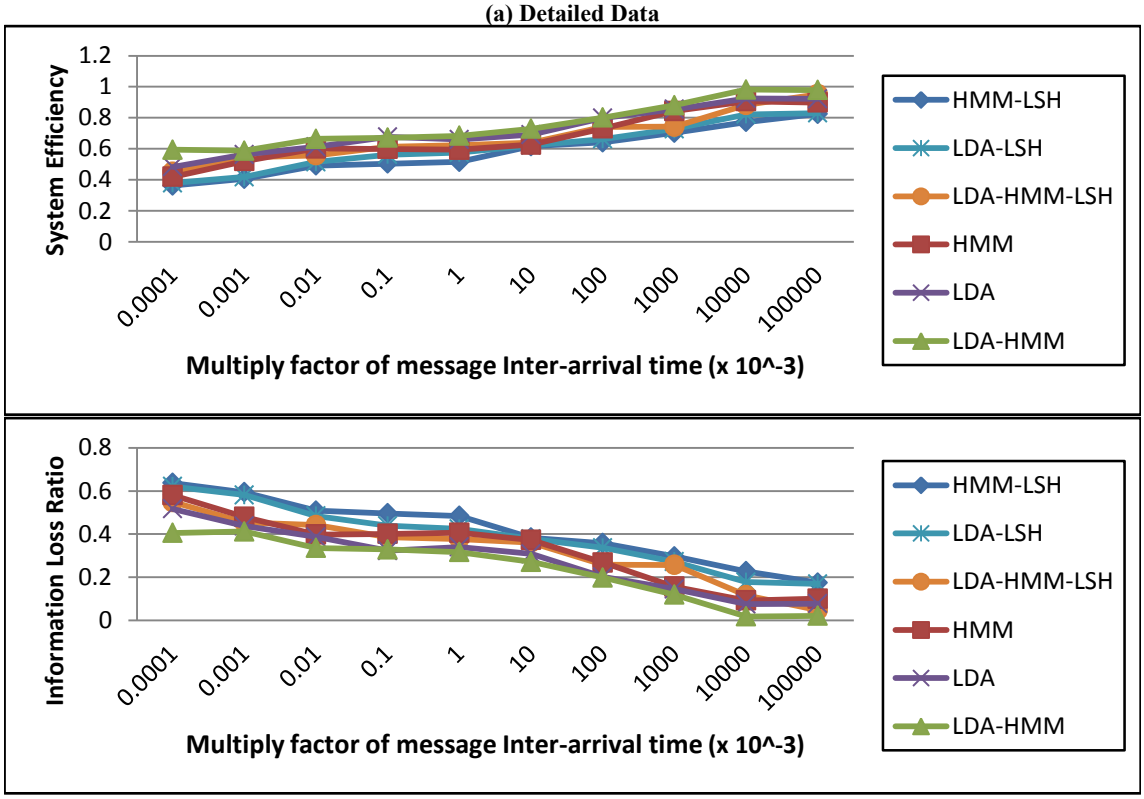
Figure 5.11 NetMem system efficiency with/without LSH at varying the number of NMemAgents

The Overall findings and conclusion

- The estimated NetMem system response time assumed 100% prediction accuracy for all reasoning models
  - This means more transferred data/semantics messages among NetMem agents and, so, there was long response time compared with the actual response time.



(a) Detailed Data



(b) Abstracted Data

Figure 5.12 NetMem system efficiency and information loss ratio with/without LSH at varying inter-arrival time of query messages

– The response time for retrieving detailed data is better than the one of abstracted data

since detailed data are most probable to be found at lower NetMem system levels with shorter-term memory.

- The hybrid semantics reasoning model with LDA-HMM technique succeeded in achieving the highest NetMem system efficiency compared with reasoning models with HMM or LDA
- Utilizing LSH results in lower NetMem system efficiency and high information loss ratio compared with the case of no LSH.
- Operation with LSH reflected decreasing in the efficiency of the NetMem system compared with the efficiency at operation without NetMem.
- There is a trade-off between NetMem system efficiency (e.g., response time) and effectiveness (e.g., accuracy of implemented semantics reasoning models).
- Reducing data dimensionality using LSH with the adoption of semantics reasoning models with low capabilities in operation efficiently with reduced dimensional data affected negatively NetMem system efficiency.

5.4.2 Results based on NetMem analytical model for data abstraction process

Based on the proposed model for computing the NetMem system processing time as illustrated in equation (5.17) and using the parameters mentioned in Table 5.1, this

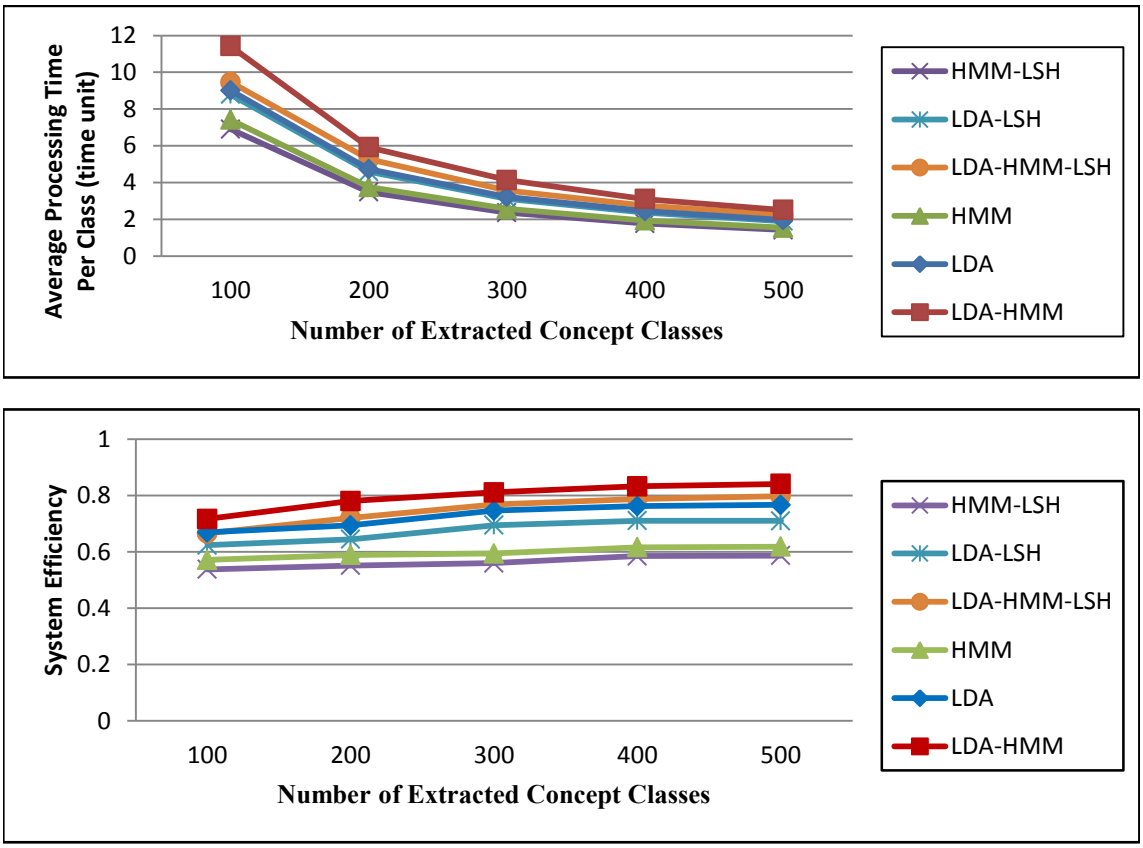


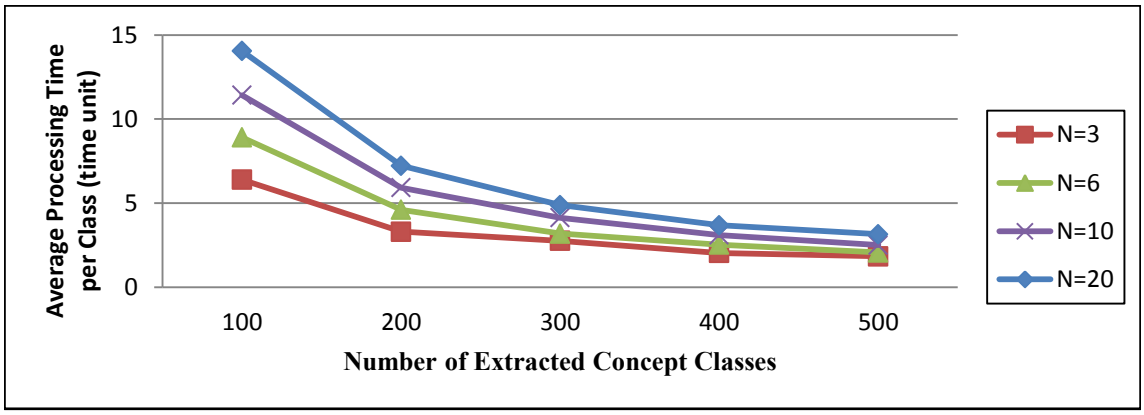
Figure 5.13 NetMem system average processing time per class and efficiency at varying number of extracted classes w.r.t. using various reasoning models with/without adopting LSH

subsection provides the results of abstraction processes for data classes with different levels of abstraction. Figure 5.13 shows the impact of varying the number of extracted classes on the NetMem system average processing time per class and efficiency at using different reasoning models.

Findings and conclusion

- Using high prediction accuracy reasoning models lengthens the average processing time per class and achieves high system efficiency.
- At not adopting LSH and w.r.t. the same used reasoning model, longer average processing per class to some extent occurs compared with the case of using LSH.
- At extracting different number of classes with different reasoning models, the higher prediction accuracy of the used reasoning models is, the longer average processing time per class will be faced.
- Even though the usage of high prediction accuracy model (LDA-HMM) lengthens the average processing time per class with/without using LSH, better NetMem system efficiency can be achieved (large number of extracted classes can be achieved in a short processing time with respect to the estimated processing time and extracted classes number. i.e., shorter average processing time per class can be achieved).
- Short average processing time per class does not infer always high system efficiency (e.g., short actual processing time per smaller number of extracted classes compared to the estimated values due to low collected data volume or low prediction accuracy).

Figure 5.14 shows the impact of having different number of NetMem levels on the average processing time and efficiency of the NetMem system at varying the number of the extracted classes with respect to different number of NetMem levels.



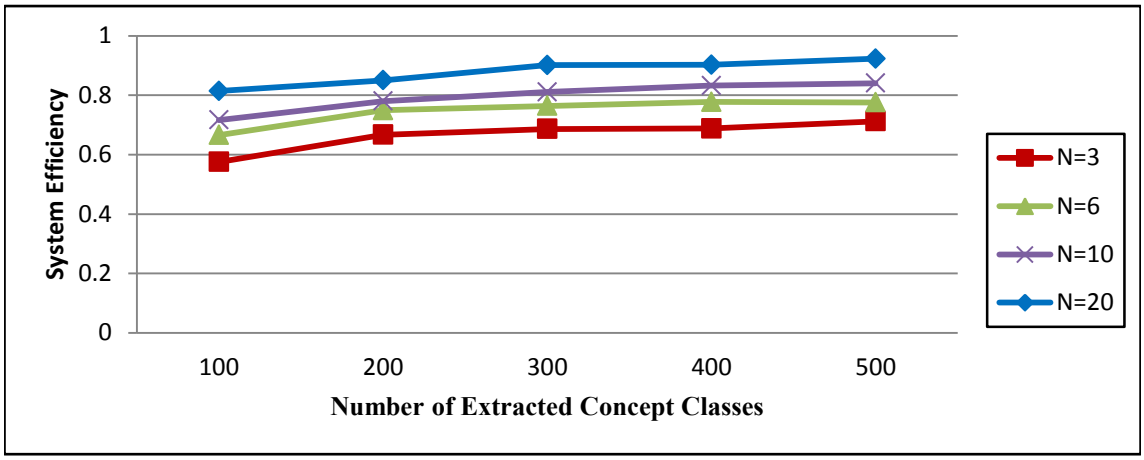


Figure 5.14 NetMem system average processing time per class and efficiency at varying number of extracted classes w.r.t. different level number

Findings and conclusion

- The larger number of NetMem levels is, the longer average processing time per class and the higher NetMem system efficiency can be achieved.
- At a certain NetMem levels number, the larger number of extracted classes is, the shorter average processing time per class and the higher efficiency can be got.
- Although increasing the number of NetMem levels lengthens the average processing time per class, higher NetMem system efficiency can be attained.
- Even though increasing the number of extracted classes infers long average processing time per class, the NetMem system efficiency achieves shorter average processing time per class at extracting large number of classes compared with the case of extracting smaller numbers.

Figure 5.15 shows the average processing time per class and the efficiency of the NetMem system at varying the inter-arrival time of data with respect to various class storage durations and at extracting different number of classes.

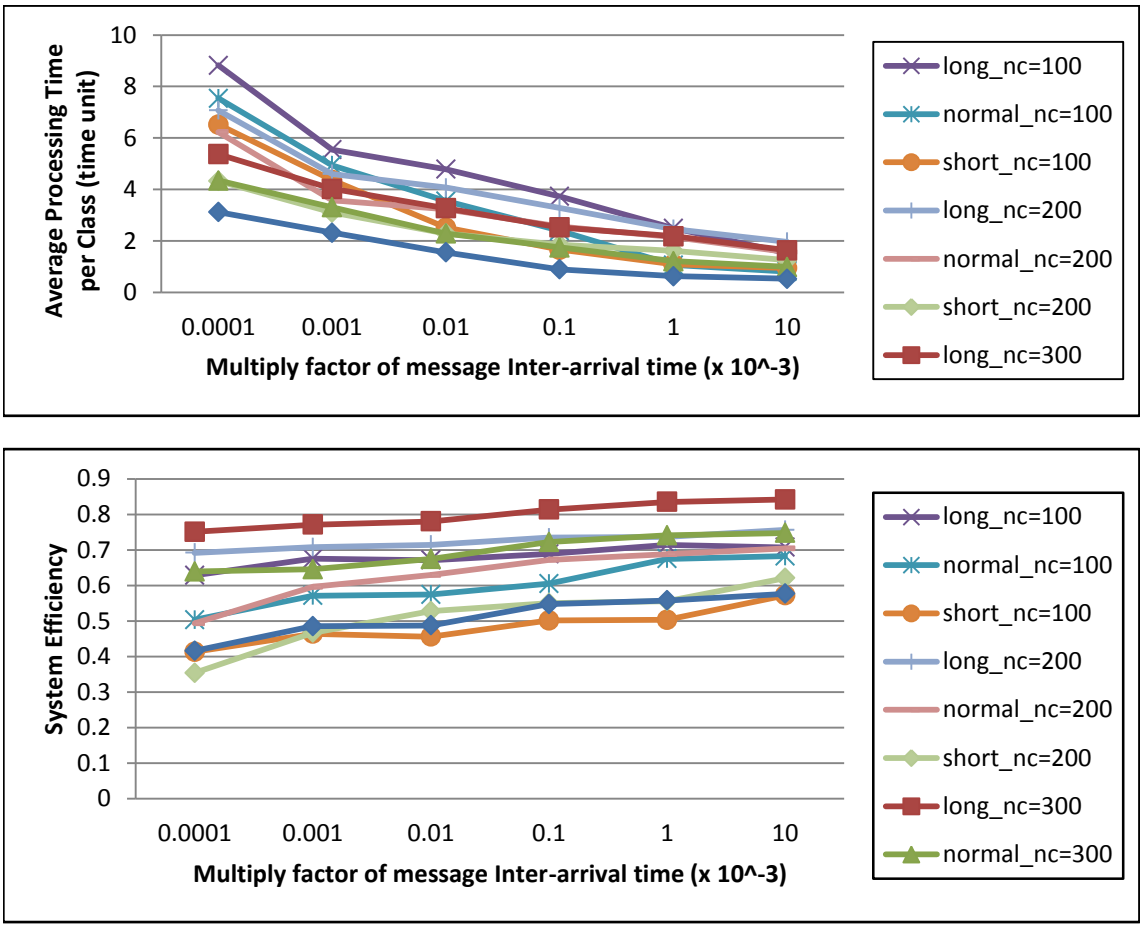


Figure 5.15 NetMem system average processing time per class and efficiency at varying data arrival rate w.r.t. different storage durations and number of extracted classes

Findings and conclusion

- Higher NetMem system efficiency at long data class storage duration in agents.
- Higher system efficiency at extracting large number of classes and low data arrival rates.
- At studying the impact of various data arrival rate with different storage durations at agents, higher NetMem system efficiency can be achieved at low data arrival rate.
- Although prolonging data/classes storage durations at agents lengthens the average processing time per class, high NetMem system efficiency can be achieved.
- Even though extracting large number of classes infers longer average processing time per class, the system achieves low average processing time per class and higher efficiency.

- In spite of having high data arrival rates which result in long average processing time per class, the adoption of high prediction accuracy reasoning model achieves high NetMem system efficiency.

Figure 5.16 shows the impact of varying the number of entities on the average processing time per class and the efficiency of the NetMem system at long storage durations and with respect to extracting different number of classes.

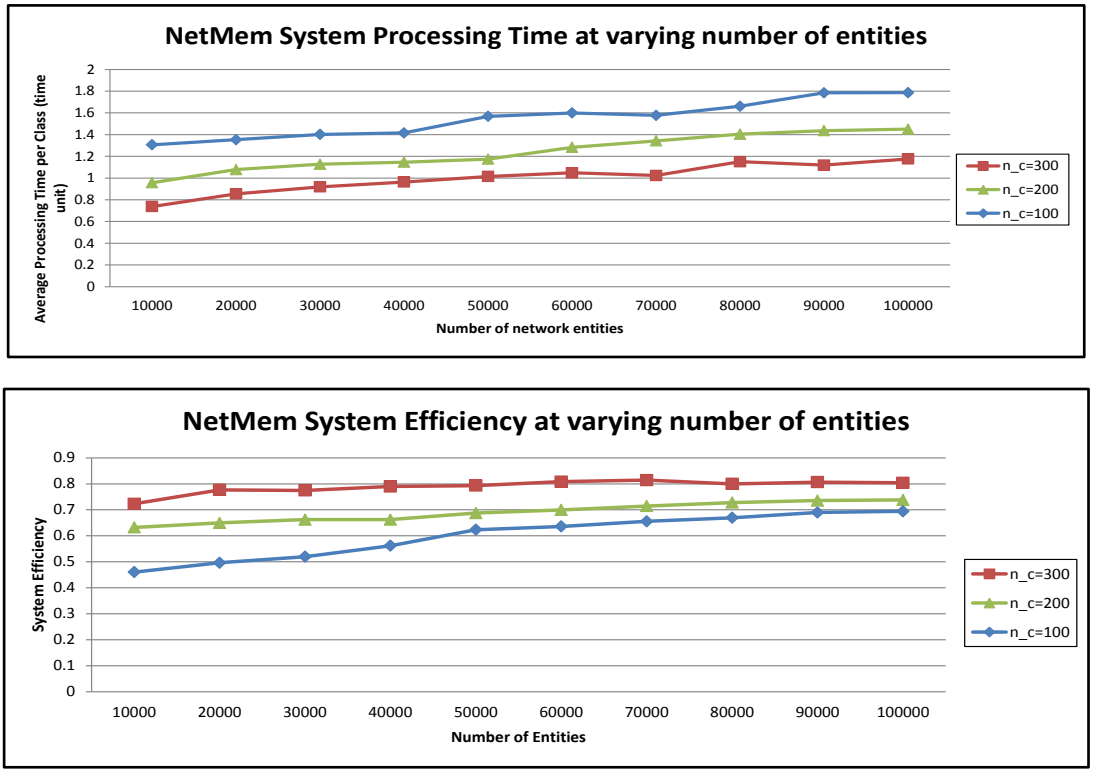


Figure 5.16 NetMem system average processing time per class and efficiency at varying number of entities w.r.t. number of extracted classes

Findings and conclusion

- Better NetMem system efficiency at having large number of entities and large number of extracted classes
- The larger number of entities is, the longer average processing time per class is obtained.
- The larger number of extracted classes, the shorter average processing time per class and higher system efficiency that can be achieved.
- Even though increasing the number of entities lengthens the average processing time per class, high NetMem system efficiency can be attained (this infers that at larger number of entities, NetMem system achieves smaller actual to estimated average processing time ratios compared with the case of small entities number).

Figure 5.17 show the impact of having different number of NetMem levels on the average processing time and efficiency of the NetMem system at varying the number of the entities.

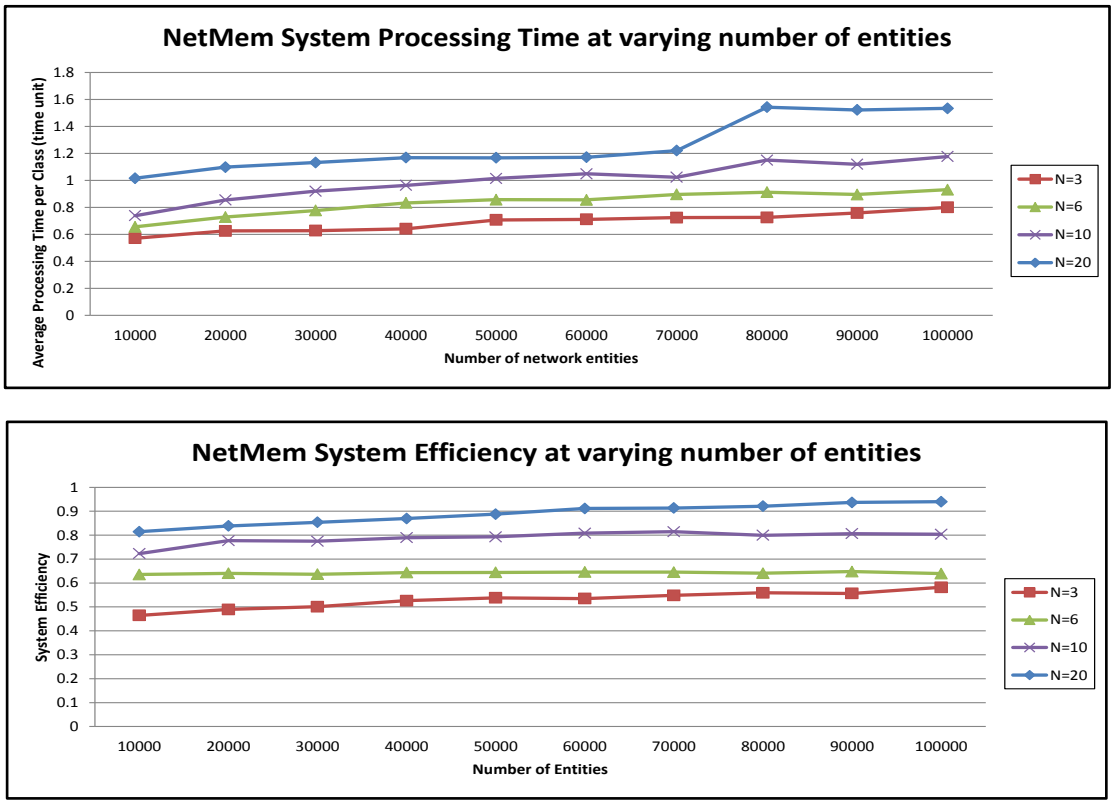


Figure 5.17 NetMem system average processing time per class and efficiency at varying number of entities w.r.t. various levels number

Findings and Conclusion

- The larger number of NetMem levels is, the longer average processing time per class and the higher NetMem system efficiency can be attained.
- More processing time is needed to build the same number of data classes in large number of NetMem levels (more agents will be found) compared with the time in case of small levels' number.
- Although increasing the number of NetMem levels lengthens the average processing time per class, higher NetMem system efficiency can be attained.
- In spite of increasing the number of entities prolongs the average processing time per class, the NetMem system efficiency achieves higher efficiency at having large number of entities compared with the case of having smaller numbers.

5.4.3 Improved NetMem Configuration

We formulated the NConOP problem as discussed in subsection 5.3.4. We investigated that optimization study because we did not find, also using simulation studies, a superior

reasoning model that can implement in NMemAgents located in NetMem levels and achieve higher NetMem efficiency without disadvantages (e.g., long processing time to some extent as in case of using LDA-HMM). We got optimal solutions for the NConOP problem through using two different optimization approaches:

1. Random Selection (Random Optimization [129])
    - Proposing random configurations for the NetMem system where it samples, depending on uniform distribution, reasoning schemes for the ***N*** NetMem levels and comprising agents to maximize NetMem efficiency. Figure 5.18 illustrates the pseudo code of the optimization algorithm with random selection. We have three different reasoning schemes ("HMM","LDA", and "LDA-HMM") which can be used.
    - Through certain number of iterations, if a sampled configuration achieves a better solution, then, this solution will be considered.

```
For(number of iterations)
        sample uniformly NetMem configuration
        value = evaluate fitness function
        if(value > max_value)
                select the configuration
                max_value = value
                if(max_value ≥ threshold_value)
                        break
End
```

Figure 5.18 Pseudo-code of the random selection algorithm for optimization

2. Genetic Algorithm (GA)
    - *Full allocation process*
        - Allocating reasoning schemes over agents in each NetMem level in order to maximize efficiency taking into consideration better solutions through GA operations.
    - *Reduced allocation process*
        - Allocating reasoning schemes over NetMem levels assuming that all level's agents adopt the same reasoning scheme (considering better solutions).

Based on the obtained comparison results through using different optimization schemes, we applied GA with reduced allocation process to search for optimal NetMem configurations. Table 5.2 shows the parameters used by the GA-based optimization problem.

Table 5.2 The parameters of the genetic algorithm-based optimization problem

Input parameter	*min value*	*max value*	*Parameter description*
Initial population size (p_{size})	10	200	Provides number of the initial proposed problem solutions
fitness value threshold	0.9	0.9	Indicates to the minimum value of fitness that should be reached
solution vector length	10	10	Describes the number of semantics reasoning models implemented at NetMem levels
iterations number	5	5	Provides the number of counts that GA will run
parent usage ratio (P_s)	0.9	0.9	The percentage of initial solutions that will used in crossover and mutation operations
Crossover ratio (P_c)	0.5	0.5	The value on which decision will be taken to perform crossover process
mutation ratio (P_m)	0.05	0.05	The threshold on which the mutation process will operate
Multiply factor for number of agents	10	50	Provides the scaled number of agents per each NetMem system level
Multiply factor for inter-arrival time ($1/\lambda$) of data messages between agents	0.1	1	Indicates to the decreasing in the arrival rates of messages to agents
Number of network entities	10000	100000	Describes the volume of communicating entities with the NetMem system

5.4.3.1Optimization problem results using different optimization schemes

Figure 5.19 and Figure 5.20 illustrate the optimization problem convergence time and the maximum obtained efficiency at the best got solutions while varying the number of population sizes at using different optimization schemes. We studied the impact of various operation parameters as large number of NetMem agents, long inter-arrival time of messages, and large number of network entities.

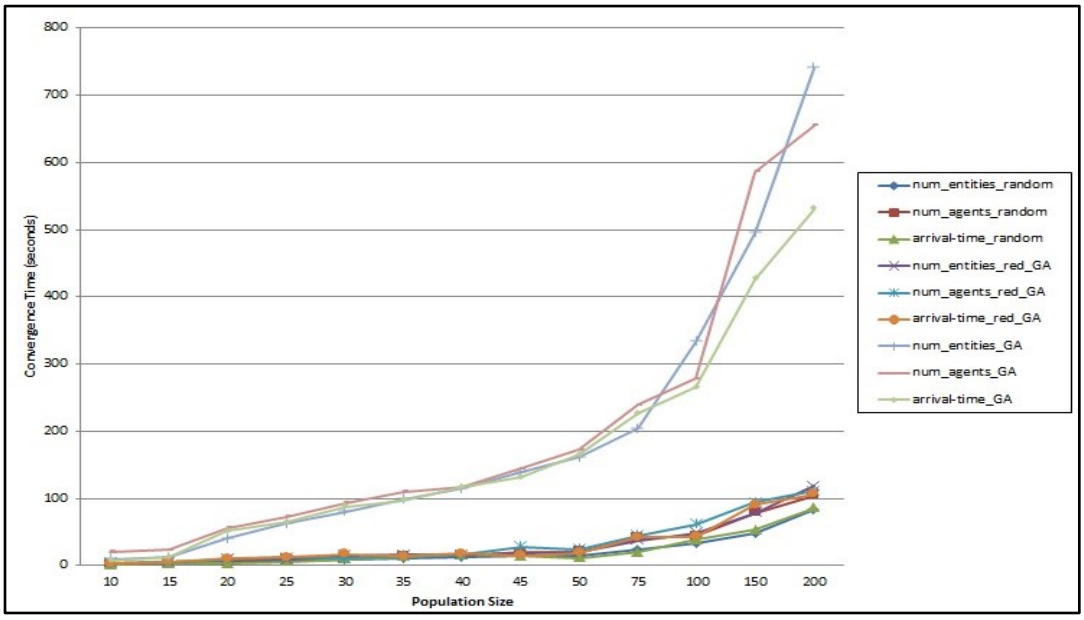


Figure 5.19 The optimization problem convergence time at using different optimization schemes

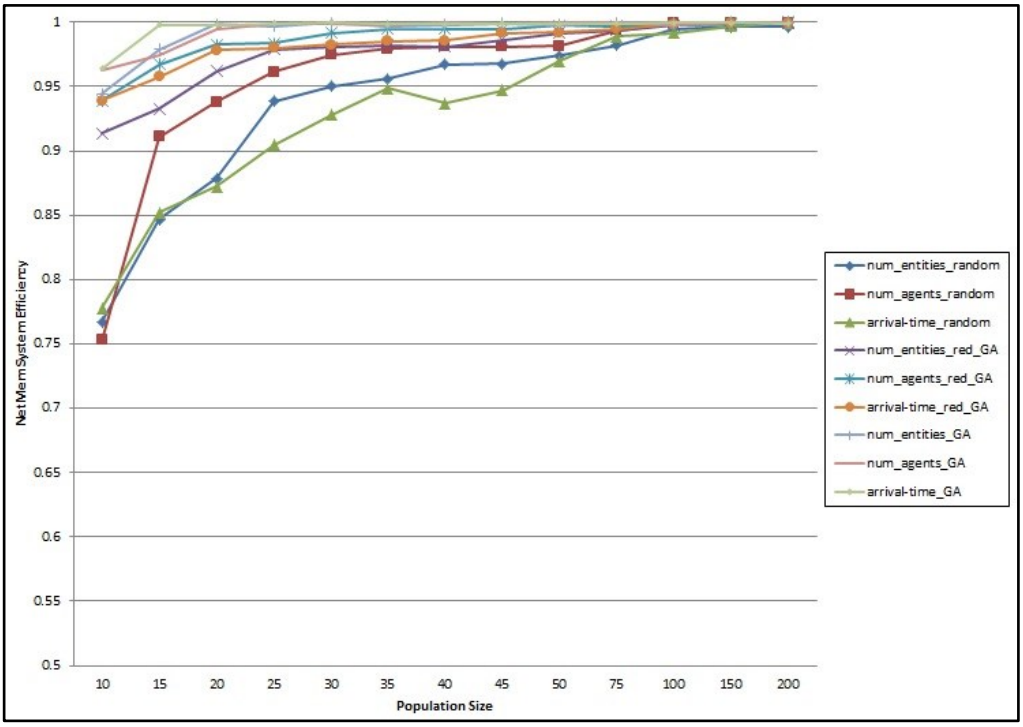


Figure 5.20 NetMem efficiency at using different optimization schemes

Findings:

- Better solutions for the optimization problem were obtained in case of using the GA with full allocation process for the reasoning schemes over NetMem agents.
- High NetMem efficiency and short convergence time at using GA with reduced allocation process at small population sizes compared with the same case at using GA with full allocation and random selection.
- Random selection with small population sizes achieved low NetMem efficiency and the shortest convergence time.
- At large population sizes, random selection and GA with reduced allocation achieved solutions with high NetMem efficiency and short convergence time
- Having limited number of reasoning schemes (i.e., three) over ten NetMem levels aids in having good solutions at using random selection with large population sizes.

5.4.3.2 Optimization problem results using the GA with reduced allocation process

From the obtained results in last subsection, good levels of NetMem efficiency were obtained using GA with reduced allocation and with low convergence time. So, we utilized GA with reduced allocation algorithm to investigate more analyses for the optimization problem with respect to the impact of many factors, such as changing the storage duration of data classes in NMemAgents. Table 5.3 illustrates the recommended NetMem configurations obtained from solving the optimization problem for optimizing NetMem efficiency. The column family of recommended solutions provides the recommended semantics reasoning models to be implemented at NMemAgents within each NeMtm level. For example, the string vector of reasoning models: **"HMM,HMM,HMM,HMM,LDA,HMM,LDA-HMM,HMM,LDA-HMM,HMM"** means that the first NetMem level will have HMM-based reasoning models in its NMemAgents and the last level will have also HMM. That configuration can aid in enhancing the NetMem efficiency to reach 0.999931543.

Table 5.3 Recommended NetMem configurations to optimize NetMem efficiency

Output parameter	Variation System Parameters	Recommended Solutions (example)	Max. NetMem System efficiency	Parameter description
Optimal Population	Small number of agents	**"HMM,HMM,HMM,HMM,LDA,HMM,LDA-HMM,HMM,LDA-HMM,HMM"** 7 HMM;1 LDA;2 LDA-HMM	0.999931543	Provides optimal configuration for the NetMem system [note: the output value is represented as vector of strings where each entry clarifies the reasoning model used per each NetMem level or entry index]
	Large number of agents	**"HMM,LDA-HMM,LDA-HMM,HMM,HMM,HMM,HMM, LDA-HMM,LDA-HMM,LDA-HMM"** 5 HMM;0 LDA;5 LDA-HMM	0.999935034	
	Long inter-arrival time of messages	**"HMM,HMM,LDA-HMM, LDA, LDA, LDA, LDA, HMM,LDA,LDA-HMM"** 3 HMM;5 LDA; 2 LDA-HMM	0.999656562	
	Short inter-arrival time of messages	**"HMM,LDA-HMM, LDA, HMM, HMM,LDA,HMM,LDA,HMM,LDA"** 5 HMM;4 LDA;1 LDA-HMM	0.999122119	
	Large number of entities	**"LDA,LDA,HMM,HMM,LDA,LDA-HMM,LDA-HMM,LDA-HMM, HMM,LDA"** 3 HMM; 5 LDA; 2 LDA-HMM	0.999502482	
	Small number of entities	**"HMM,HMM,LDA-HMM, LDA, HMM,HMM,LDA-HMM, HMM, LDA, HMM"** 6 HMM;2 LDA;2 LDA-HMM	0.99903063	

Output parameter	Variation System Parameters	Convergence time at recommended solutions		Parameter description
		Min (seconds)	Max (seconds)	
Convergence Time	number of agents	3.723	111.197	Indicates to the time duration of finding optimal configuration (i.e., reaching maximum NetMem efficiency)
	inter-arrival time of messages	3.161	106.372	
	number of entities	3.157	118.077	

5.4.3.3 The results of optimizing NetMem configuration based on utilizing optimal solutions

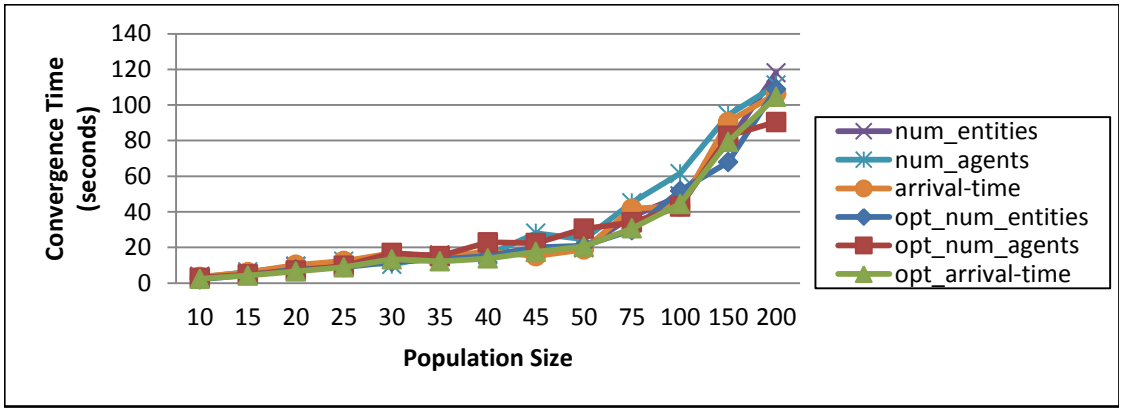
We run the GA with reduced allocation to solve our optimization problem using initial population of solutions that were obtained from the previous optimization results in last sections. In other words, we assumed that the GA will have optimized solutions, instead of random input solutions, in the initial phase in order to investigate more enhanced NetMem configurations (i.e., recommended reasoning models at NMemAgents to improve NetMem efficiency) at shorter convergence time and at small population sizes. Table 5.4 shows the results of solving the optimization problem when input optimal candidate populations. The comparison between using optimal and random initial populations, at varying number of entities, agents and inter-arrival message time, shows that higher NetMem efficiency was reached at shorter convergence times.

Table 5.4 Optimum NetMem configurations at input optimal candidate populations

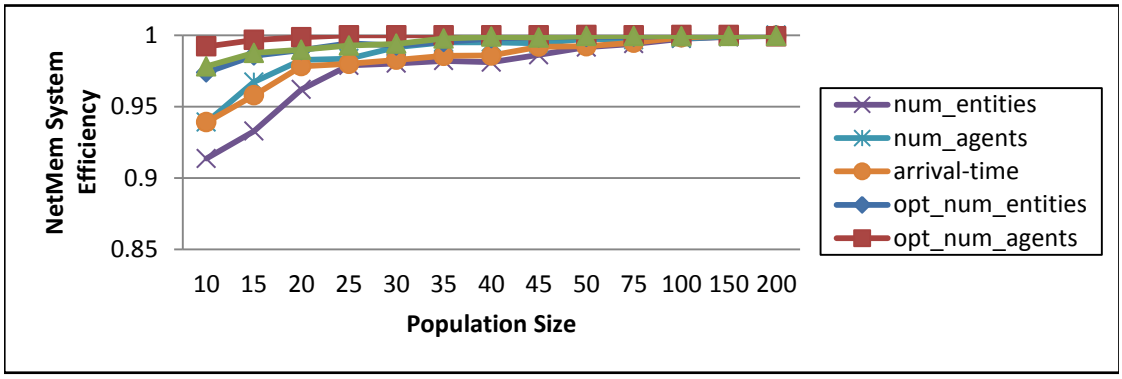
Output parameter	*Variation System Parameters*	*Input Solutions (example)*	*Max. NetMem System efficiency*	*Parameter description*
Optimal Population	Large number of agents	**"HMM,LDA-HMM,LDA-HMM,HMM,HMM,HMM,HMM, LDA-HMM,LDA-HMM,LDA-HMM"** 5 HMM;0 LDA;5 LDA-HMM	0.999116283	Provides optimal configuration for the NetMem system [note: the output value is represented as vector of strings where each entry clarifies the reasoning model used per each NetMem level or entry index]
	Long inter-arrival time of messages	**"HMM,HMM,LDA-HMM, LDA, LDA, LDA, LDA, HMM,LDA,LDA-HMM"** 3 HMM;5 LDA; 2 LDA-HMM	0.999364948	
	Large number of entities	**"LDA,LDA,HMM,HMM,LDA,LDA-HMM,LDA-HMM,LDA-HMM, HMM,LDA"** 3 HMM; 5 LDA; 2 LDA-HMM	0.9939491461546882	

Output parameter	*Variation System Parameters*	*Convergence time at recommended solutions*		*Parameter* description
		Min. in seconds *Opt. input (Random input)*	***Max in seconds*** *Opt. input (Random input)*	
Convergence Time	number of agents	2.902 (3.723)	90.432 (111.197)	Indicates to the time duration of finding optimal configuration (i.e., reaching maximum NetMem efficiency)
	inter-arrival time of messages	2.38 (3.161)	104.548 (106.372)	
	number of entities	2.05 (3.157)	108.984 (118.077)	

A comparison between utilizing random and optimal populations at the initial operation phase of the GA to solve the optimization problem is depicted in Figure 5.21. Higher NetMem system efficiency was achieved at using smaller initial population sizes of optimized solutions compared with same sizes with random solutions. Additionally, in case of inputting initial optimized solutions, high NetMem efficiency was achieved in quicker time compared with the case of initial random solutions.



(a) The convergence time versus the population size



(b) NetMem system efficiency versus the population size

Figure 5.21 The optimization problem convergence time and NetMem efficiency at using random and optimal populations

Findings and conclusion

- Reaching optimum configuration with high NetMem efficiency and short convergence time at using small initial population sizes and varying number of agents.
- Getting better solutions for optimum NetMem system configuration with high efficiency after short convergence times and at having small population sizes.
- We approximately had optimized solutions (i.e., optimal NetMem system configuration with high efficiency) after same time duration while changing various NetMem system parameters especially at small population sizes.

- Optimized NetMem configuration can be executed by formulating a NP-hard optimization problem for maximizing NetMem efficiency using heuristic algorithms, such as genetic algorithm.
- Input optimized solutions for the NetMem system configuration optimization problem enhanced the obtained efficiency and diminished the convergence time of solving the problem.

5.5 Conclusion

In this chapter, we provided a formal representation model for NetMem. Also, we developed an analytical model for studying the efficiency of NetMem when it is designed and operated as a multi-agent distributed system. The analytical study provided measures for data transfer overhead and response time in case of data retrieval processes at using various semantics reasoning models and at varying different operation parameters, such as number of network entities, number of NMemAgents and the inter-arrival time of query messages received by agents. Also, we calculated NetMem processing time in case of data abstraction processes with respect to using various reasoning models. The obtained results by the analytical study showed the efficiency of NetMem to work with diverse reasoning models with full- and reduced-dimensional data. The LDA-HMM hybrid semantics reasoning model succeeded in achieving the highest NetMem system efficiency compared with reasoning adopting HMM or LDA with/without using LSH. Additionally, there was a trade-off between NetMem system efficiency and enhancing the space complexity of the implemented semantic reasoning models at using LSH.

Also, an optimization methodology for reaching near-optimum NetMem configuration was presented and solved using the genetic algorithm. The results of the optimization provided recommendations for implementing specific reasoning algorithms at NMemAgents within NetMem levels to improve the overall NetMem system efficiency.

Chapter 6

6 NETMEM EFFECTIVENES

6.1 Overview

For the purpose of evaluating the effectiveness (e.g., accuracy and recall) of NetMem, we conduct a set of simulation scenarios. NetMem is designed as one NMemAgent and implemented as an application written in Java including Java classes for the operations and interactions of NMemAgent main components. We implemented a small-scale prototype for NetMem and we compared NetMem effectiveness with Snort [130] over low and high volume of real Internet traffic data. Additionally, we run simulation scenarios via integrating NetMem Java classes with java-based network simulator (J-Sim) [131, 132]. A simulation scenario for a network of 50 nodes is conducted, with the aid of real offline KDD datasets [133], to investigate the impact of learning concepts related to file transfer services and communication protocols over improving QoS of running services on top of TCP and UDP protocols. Our NetMem simulation studies include implementation of different semantics reasoning models (using simple statistical-analysis-based, LDA-based, HMM-based, and the hybrid LDA-HMM-based models).

6.2 NetMem Prototype

A small-scale NetMem prototype is built as shown in Figure 6.1. A practical and simple network of five entities was tested. Those entities connect to the Internet and they were assigned IP addresses as illustrated in the figure. There were two hosts and two servers. FTP and Web servers were implemented over two static laptops running Windows 7.

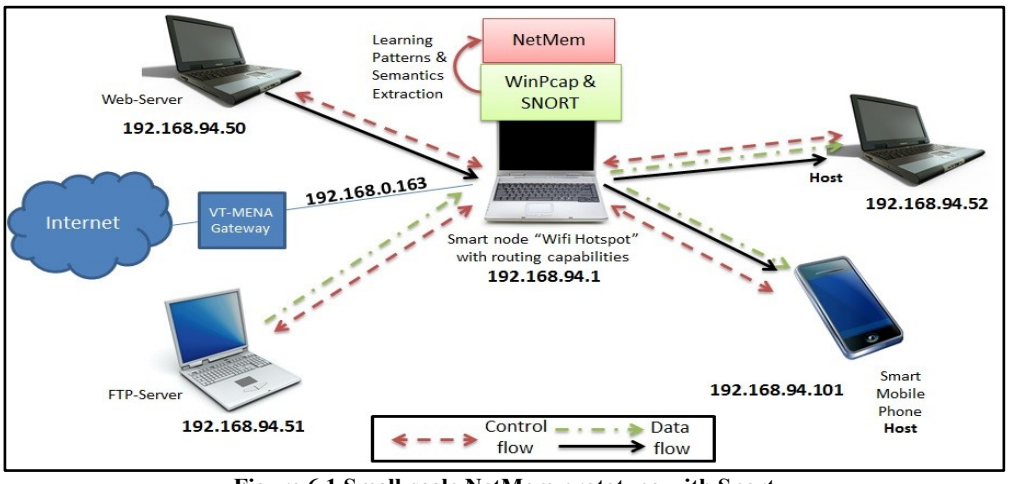


Figure 6.1 Small-scale NetMem prototype with Snort

Two hosts were built over another two static laptops to handle data from servers. NetMem was implemented as one NMemAgent.

NMemAgent was mounted over a Windows 7 laptop, an entity with routing functionalities that can capture data/control packets going from/to hosts to/from servers and Internet. NetMem was designed as an application written in Java for the operations of NMemAgent entities: NCI, DVA, memory and SM. NetMem operation depended on low and high volume of real-Internet traffic data (the low volume is in range of 10 GB and the high volume is in range of 60 GB) collected via snort [130]. Snort is an application that can capture raw real traffic data packets via its data packet capturing tool "Winpcap" [134]. Snort has different modes of operation. Snort is an open source networking tool that can work as data sniffer, data packets logger or an intrusion detection and prevention system. Snort operates based on a defined set of rules.

Based on collected data packets via snort, NetMem represented data in the memory with/without using LSH with various lengths of hash functions. NetMem effectiveness was tested at using various semantic reasoning models: simple statistical-analysis-based model, LDA-based model, HMM-based model and the LDA-HMM-based model. The reasoning model implemented by LDA-HMM algorithm is called the hybrid model or HIT. NetMem reasoning models were designed to reason about semantics related to abnormal/normal behavior of file transfer services (i.e., application concern) using TCP and UDP protocols (i.e., communication concern). NetMem effectiveness was compared with snort in correctly detecting and learning normal and abnormal behavior classes of services' data flows.

We targeted the following metrics for evaluating the performance as shown in Table 6.1: processing overhead, space complexity, prediction accuracy, recall, precision, and false negative and positive ratios. We compared the performance of our system with various reasoning models versus snort. We evaluated NetMem system performance without LSH and with LSH at various hash function lengths $\boldsymbol{k}$ values ($\boldsymbol{k}$ = 3, 4 and 5). The $\boldsymbol{k}$ value refers to the reduced number of attributes in reduced-dimension data profiles.

Table 6.1 Prediction analysis metrics and equations

Metric	Function	Equation	Unit
Processing Overhead (T)	Measures the time complexity of the implemented semantics reasoning model for detecting and learning all behavior classes of running service flows	$T = T_R$ where T_R is the time-overhead caused by the implemented algorithms in the semantics reasoning model	min
Space Complexity (S)	Identifies the storage space required for maintained data that will be used by the implemented semantics reasoning model to learn all behavior classes of service	S is memory_size where memory_size is the storage space of the working memory for	Kbytes

	flows	learning data patterns	
Prediction Accuracy (A_c)	Calculates ratio of true positive (Tp) classes and true negative (Tn) classes that are learned with respect to all behavior classes (N_c) that should be learned	A_c =(Tp+Tn) / N_c	---
False Positive (Fp) Ratio	Calculates ratio of normal behavior classes that misclassified to abnormal classes according to N_c	Fp / N_c	---
False Negative (Fn) Ratio	Calculates ratio of abnormal behavior classes that misclassified to normal classes according to N_c	Fn / N_c	---
Recall (R)	Measures the effectiveness of system to learn abnormal behavior classes (including classes of attacks) with respect to Fn	R= Tp/(Fn+Tp)	---
Precision (P)	Measures the efficiency of system to learn abnormal behavior classes (including classes of attacks) with respect to Fp	P= Tp/(Fp+Tp)	---

6.2.1 Results over low and high volume data

Figure 6.2 illustrates the prediction accuracy of the NetMem system, using various semantics reasoning models, and snort at learning patterns of low and high volume of real Internet traffic data.

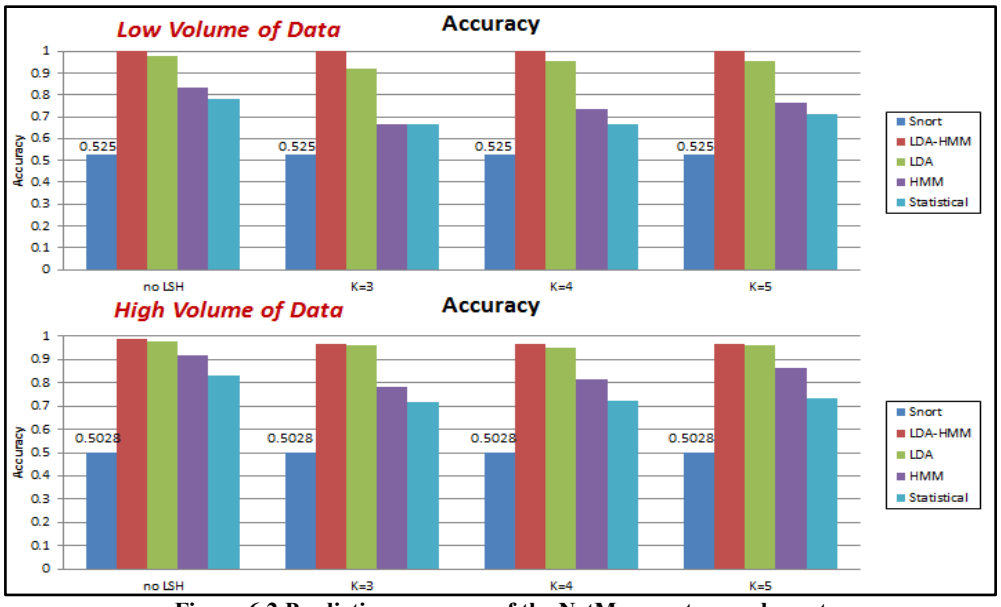


Figure 6.2 Prediction accuracy of the NetMem system and snort

Figure 6.3 depicts the false negative rate of the NetMem system, using various semantics reasoning models, and snort at learning patterns of low and high volume of real Internet traffic data.

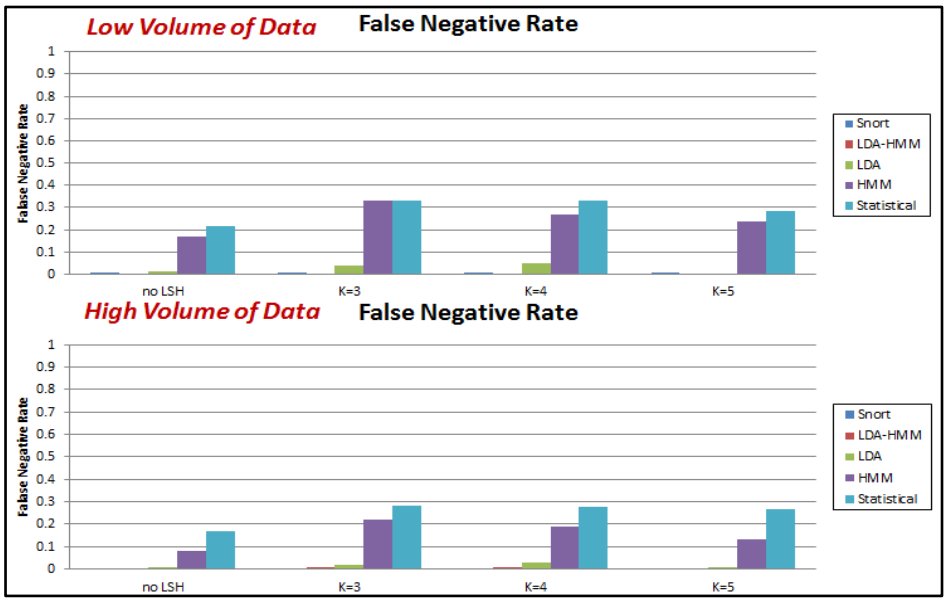


Figure 6.3 False negative rate of the NetMem system and snort

Figure 6.4 shows the false positive rate of the NetMem system, using various semantics reasoning models, and snort at learning patterns of low and high volume of real Internet traffic data.

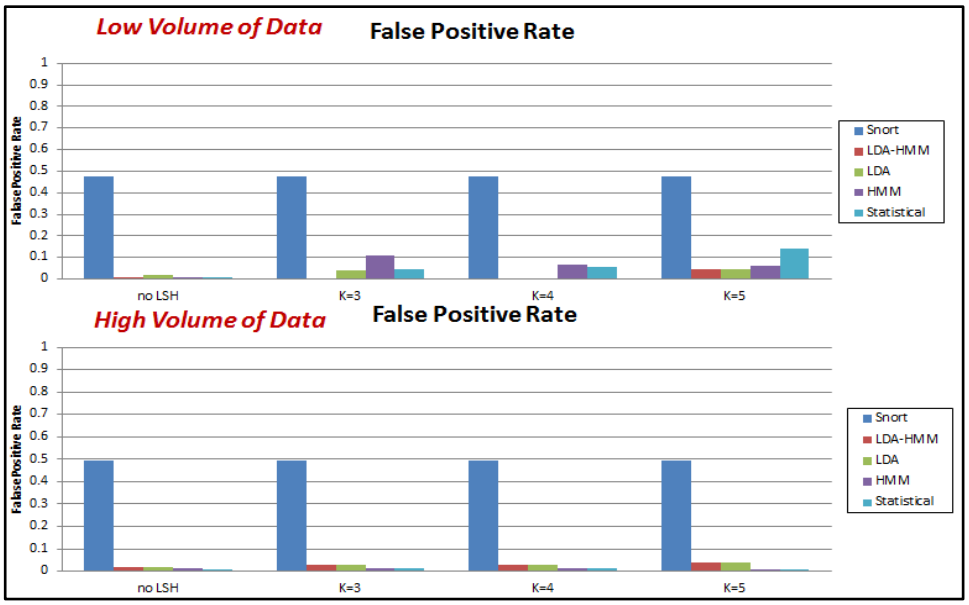


Figure 6.4 False positive rate of the NetMem system and snort

The worst false positive rate was attained by snort (i.e., snort was not able to learn correctly normal behavior classes of running flows). Snort can be considered a stateless firewall in general [135]. As an example, a stateless firewall will prevent legitimate

packets concerning a FTP session from passing since it has no knowledge that those packets destined to a protected network adopt a certain host's destination port, such as 4970.

Figure 6.5 shows the recall of the NetMem system, using various semantics reasoning models, and snort at learning patterns of low and high volume of real Internet traffic data.

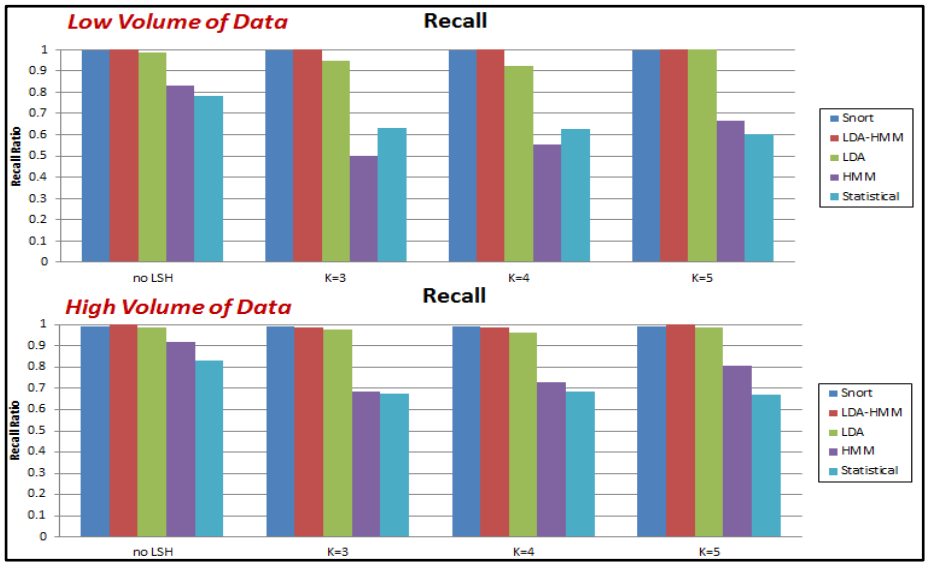


Figure 6.5 The recall of the NetMem system and snort

Figure 6.6 shows the precision of the NetMem system, using various semantics reasoning models, and snort at learning patterns of low and high volume of real Internet traffic data.

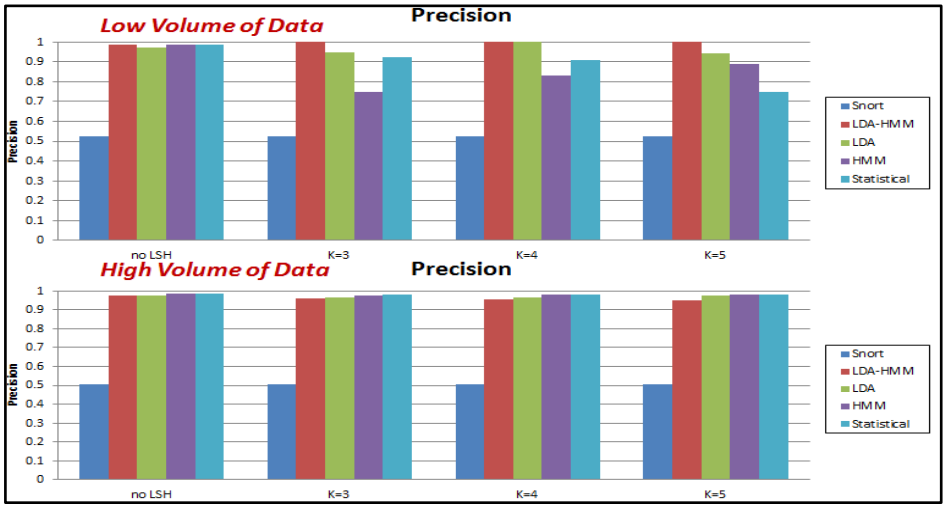


Figure 6.6 The precision of the NetMem system and snort

Figure 6.7 shows the processing time overhead of the NetMem system, using various semantics reasoning models, and snort at learning patterns of low and high volume of real Internet traffic data.

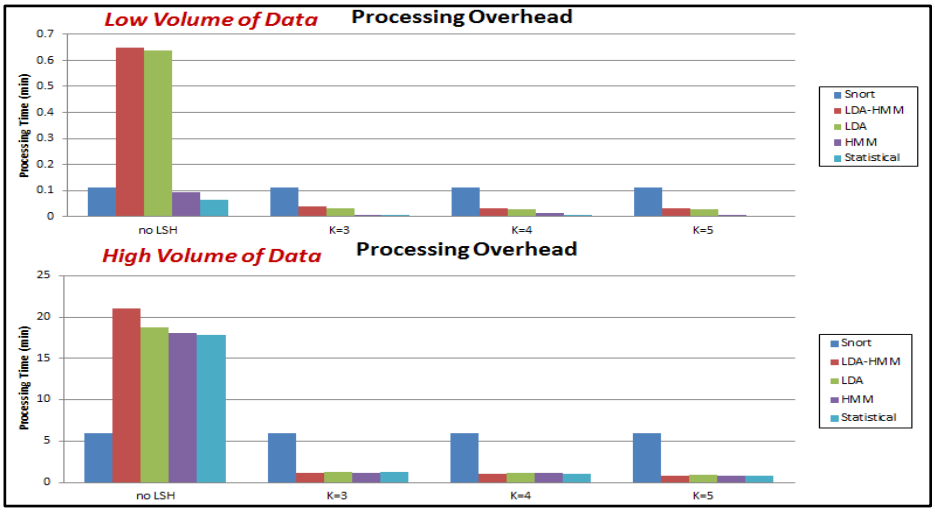


Figure 6.7 The processing time overhead of the NetMem system and snort

Figure 6.8 and Figure 6.9 show the space complexity of the NetMem system, using various semantics reasoning models, to learn patterns of low and high volume of real Internet traffic data, respectively.

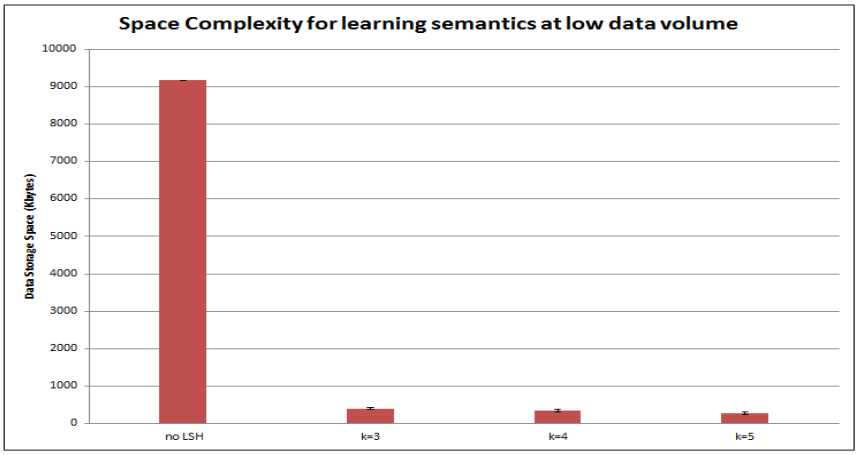


Figure 6.8 Space complexity of NetMem reasoning algorithms at low data volume

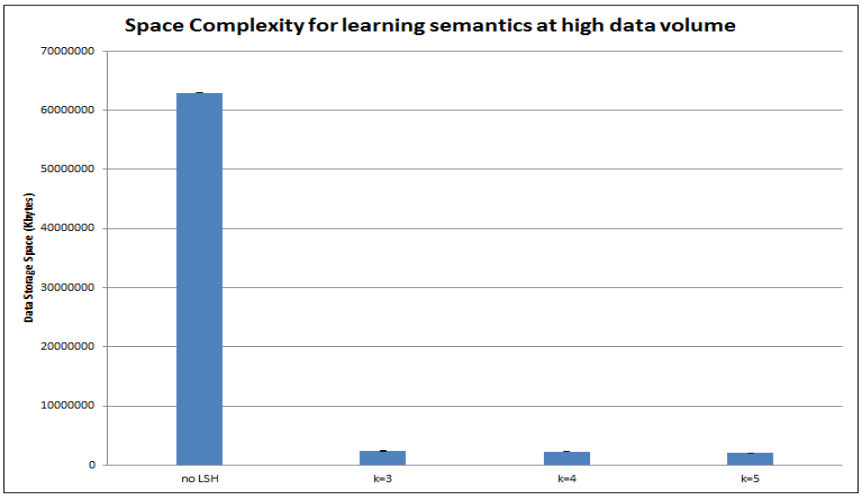


Figure 6.9 Space complexity of NetMem reasoning algorithms at high data volume

Figure 6.10 depicts the storage space saving in NMemAgent's storage memory due to the usage of LSH algorithms. Figure 6.11 and Figure 6.12 illustrate the impact of space complexity on prediction accuracy of different NetMem reasoning models over small and large data volume, respectively.

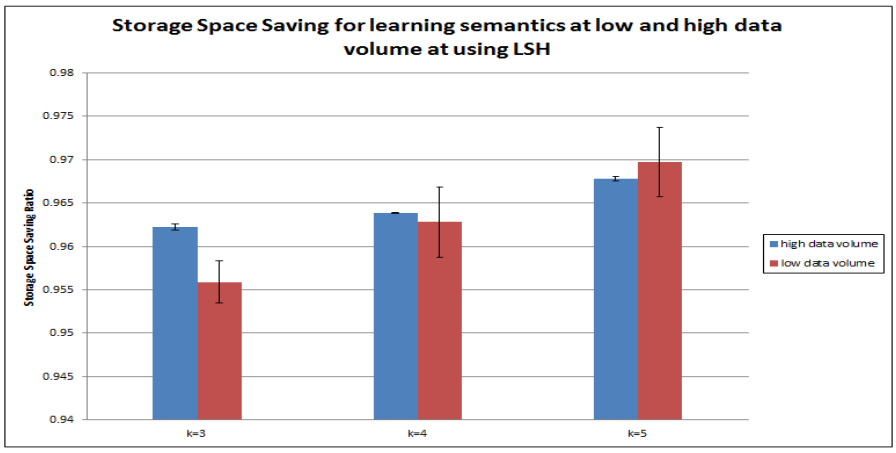


Figure 6.10 Storage space saving ratio in NMemAgent memory at using LSH

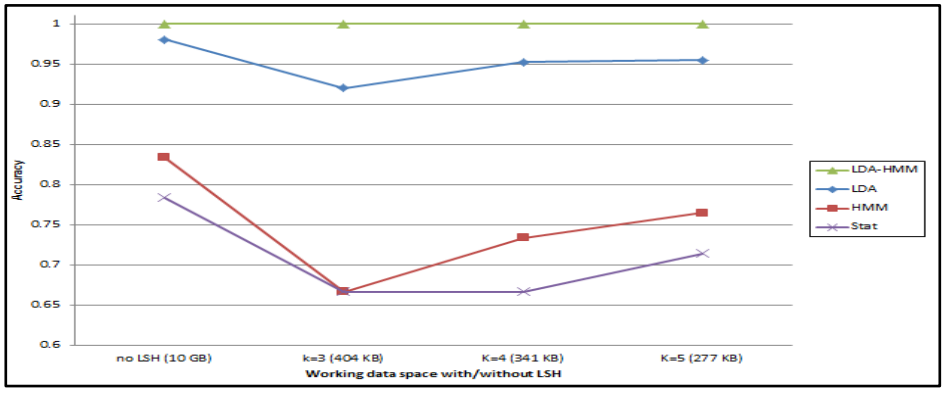


Figure 6.11 Prediction accuracy versus space complexity at low data volume

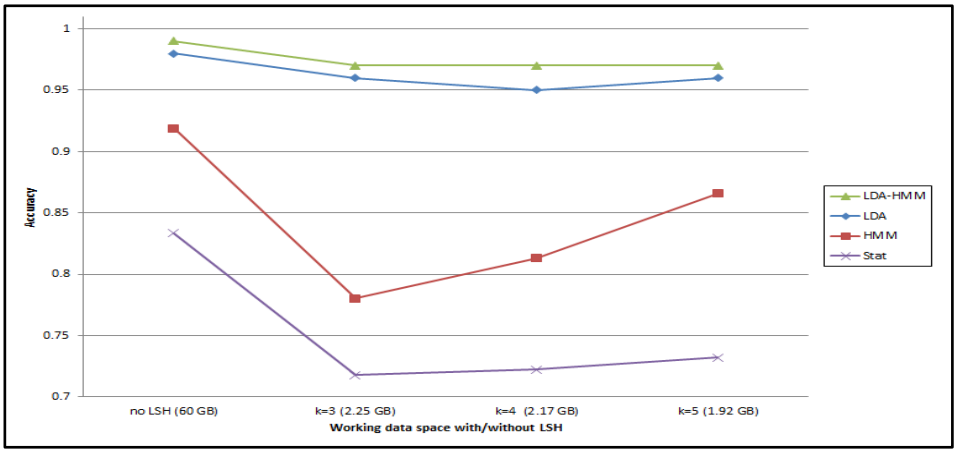


Figure 6.12 Prediction accuracy versus space complexity at high data volume

Overall findings

- Some monolithic semantics reasoning models (HMM) were not be able to learn correctly behavior semantics of service data flows (HMM does not possess capabilities to reason about high-level or latent data features).
- The hybrid semantics reasoning model (the one which combines HMM and LDA) achieved the best NetMem accuracy with/without using LSH.
- Adoption LSH reduces the dimension of data to be analyzed, and accordingly it minimizes NetMem latency.
- Although the usage of LSH resulted in losing some information (due to data dimension reduction), however, employing directed LSH (selection of data attributes or features is biased towards certain network concerns or issues) achieved good levels of NetMem efficiency (high accuracy).
- Semantics reasoning models within NetMem, with data dimension reduction, resulted in more processing time overhead compared with snort.
- The hybrid model combines two reasoning models, however, it achieved good level of NetMem latency compared with other monolithic models (the training phase of some monolithic models, at large number of data features at high volume of data, affected negatively the timeliness of those models).

Overall conclusion

- Storing reduced-dimensional data with directed features or attributes can help in having efficient semantics management systems (high accuracy and low processing time overhead).
- There is a trade-off between having a semantics management system with good levels of operation accuracy and low processing time overhead.
- NetMem can provide an effective and efficient capability for learning behavior classes of normal/abnormal network-data traffic relying on learning patterns of full- or reduced-dimensionality profiles of traffic data.
- Adopting light-weight simple statistical-based reasoning model achieves good levels of prediction accuracy with low overhead, however, lower than the LDA-based and HMM-based models.
- Implementing a LDA-based semantics reasoning model with capabilities of extracting high-level features improves NetMem performance (e.g., accuracy and false negatives) compared with other two reasoning models.
- HMM-based reasoning model integrated with unsupervised learning algorithm succeeds in extracting group of associated concept classes with accuracy and low computation overhead.
- Integrating two semantics reasoning models, with various capabilities, to form a hybrid model might lead to enhancements in learning high-level features and

improving accordingly the performance (e.g., accuracy) of semantics management systems.

- NetMem can be attached with networking tools as intrusion detection and prevention systems (e.g., snort) to enhance their effectiveness and efficiency.

6.3 NetMem with CAIDA Database

We implemented NetMem as one NMemAgent employing a reasoning model (e.g., LDA-based reasoning model) for learning patterns and extracting features over large-scale data. Here is the description of the study:

- Read data from a database file by CAIDA [11, 136] (we tested a file of size 63221 bytes which maintains information about different autonomous entities as corporations and universities).
- Each line is the database file was represented by DVA as a data profile comprises group of attributes and related values.
- Reducing data dimensionality (i.e., dimension of data profiles) and storage space in memory using LSH and assigning topics (i.e., classifying data at the file) for these data using LDA implemented in SM.
- We compare operation with/without LSH for identifying topics of file lines.

Table 6.2 shows the obtained results.

Table 6.2 NetMem results over CAIDA database file

Parameter	With LSH	Without LSH
memory size (bytes)	32671	59153
SM time (seconds)	382.53	17726.059
LSH time (seconds)	0.382	-----
Reasoning time (seconds)	69.387	17455.009
Reasoning model Perplexity	1.1389519476744725	1.00012046551213
# of extracted features	21	3356
# of analyzed profiles	953	1026
Accuracy	74.17%	60.72%

Conclusion

- NetMem was able to minimize resource consumption and processing time overhead through reducing data profiles' dimension using LSH.
- Extracting information of reduced-dimensional data might enhance the accuracy of reasoning algorithms especially when using LSH algorithms with directed selection for interesting attributes in forming reduced-dimensional profiles.

6.4 Impact of NetMem on Networking Operations

We conducted set of simulation scenarios [137, 138] using J-Sim [131, 132] to study the efficacy of NetMem to enhance networking operations in terms of improving QoS of running services as well as learning normal/abnormal behavior of those services and detecting attacks. J-Sim is a discrete-time event-driven component-based simulator based on autonomous component architecture. The behavior of J-Sim components are defined in terms of contracts. Those components can be plugged to the J-Sim during the execution process. The following subsections discuss scenarios regarding small- and large-scale NetMem-based networks.

6.4.1 Small- and Large-scale NetMem-based Network Simulation Scenarios

6.4.1.1 Small-scale Scenario I

We implemented a simple scenario [138] for a network composed of eight hosts, two nodes with routing functionalities "routers", and a shared NetMem system represented as one NMemAgent comprising entities for NCI, DVA, memory and SM. NetMem system was implemented by Java for handling data from sources and semantics request/response messages from nodes, profiles' building, feature extraction and classification and the used ARL algorithm and HMM model. In this scenario, we designed an HMM model for extracting semantics related to abnormal profiles and attacks. A simple communication protocol was built to enable interaction between NetMem and other nodes. Our scenario goal, basically, is to show the capability of NetMem for understanding at runtime and on-demand networking contexts (e.g., running services and their time-varying requirements) via learning and deriving semantics related to a) normal/abnormal flows of running services; and b) attacks and use those semantics for predictive operation to enhance QoS of running services and to strengthen security by anomalies detection. We have two different service classes for data transfer where the first service class uses TCP transport protocol between two hosts and the other service class uses file transfer protocols on top of UDP among other two hosts. We build a static route for each service class where intermediate routers transfer data packets of both services. The other four hosts are attackers (i.e., non-legitimate entities) which generate abnormal TCP/UDP flows to degrade services' QoS of legal entities. Table 6.3 shows simulation parameters.

Legal entities in the scenario can access NetMem at runtime and on-demand to store/discover/retrieve data and to learn semantics, or associated concepts, concerning their interesting services. For instance, the TCP and UDP services are provided in a

specific region and at certain periods of time through the day. Data profiles regarding traffic of those services are stored in NetMem by DVA in memory. Those profiles show source/destination IP, service type, port number, packet size and type, allocated buffer size and bandwidth, and service duration. Profiles in memory exhibit patterns that can be learnt by SM to derive semantics related to those services. Communicating entities, such as routers, can know via a) data profiles which clarify IP attributes; and b) TCP/UDP semantics: requirements of running services, and the normal and abnormal behavior of services. For example, NetMem has semantics for the TCP service which reveal that this service, within range of IPs, uses specific resources at certain time period and at particular area because of the impact of another service (i.e., UDP service).

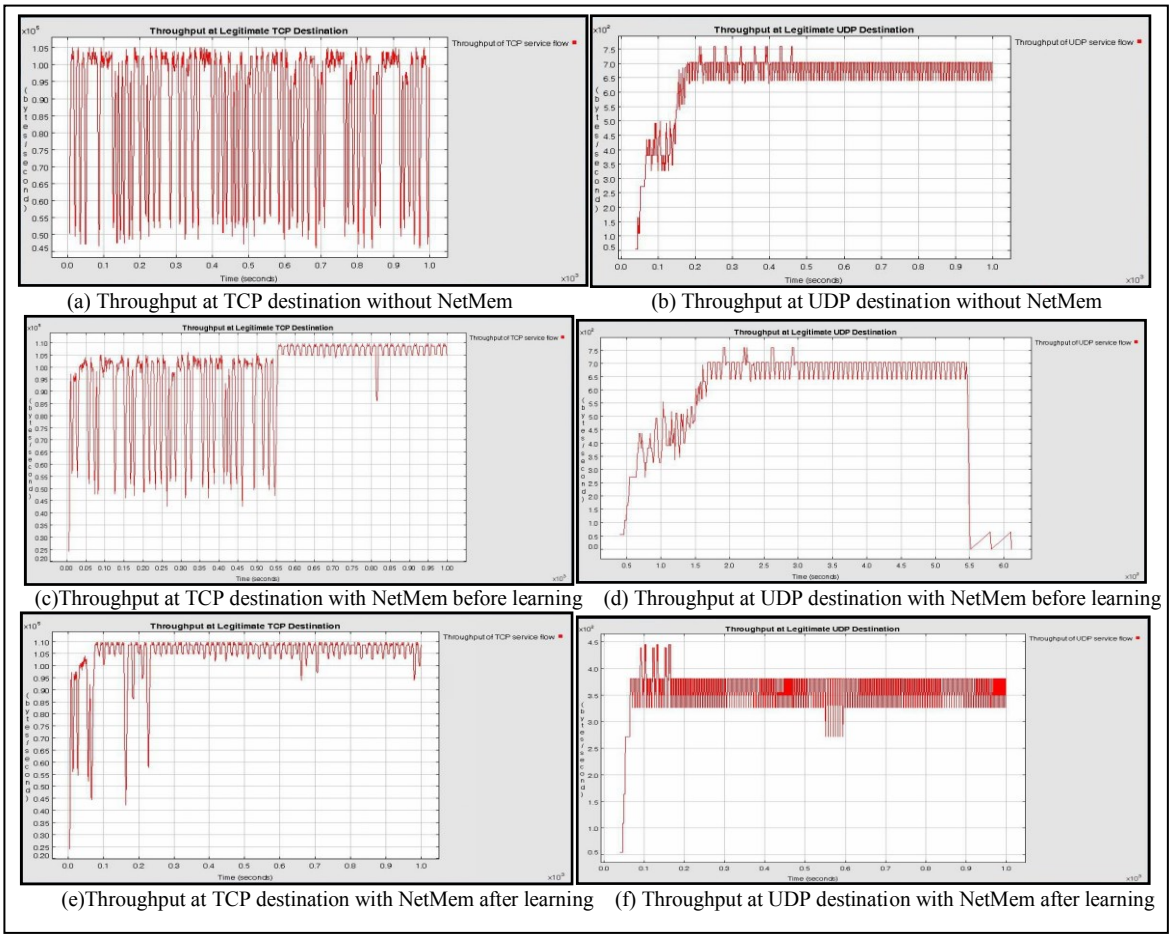
Table 6.3 Simulation parameters of the second small-scale NetMem-based network scenario

Parameter	***Value***
Link Data Rate (fixed)	1 Mbps
Router Buffer Size (fixed)	7000 packets
TCP MSS (fixed) Normal/Abnormal Flow (1) & (2)	512 /1024 & 2048 bytes
TCP MCWS (fixed) Normal/Abnormal Flow	128 bytes
TCP Time to Live (fixed) Normal/Abnormal Flow	255 seconds
UDP Client/Reply Timeout	30 seconds
UDP Packet Size (fixed) Normal/Abnormal Flow (1) & (2)	512/ 512 & 2048 bytes
Start Time: Normal TCP flow/two abnormal TCP flows/Normal UDP flow/two abnormal UDP flows	From beginning/form beginning/40 seconds/90 and 100 seconds
Propagation Delay (fixed)	100 mseconds
Rate of Using Sensory Functions in Hosts for Recognition Attack Alerts (variable)	Every 60 seconds
Rate of NetMem Access and Data Patterns Detection in StM by SM (variable)	Every 100 seconds
Rate of Change for Contents (i.e. Data Patterns) in memory	Every time Hosts send/receive data
HMM approach/number of training sequences	Unsupervised using Baum-Welch algorithm/ 1000
Simulation Time	1000 seconds

To derive semantics, SM analyzes the different profiles in memory where it learns group of attributes (e.g., packet_size) in profiles using our designed ARL algorithm. After that, SM discovers features per each profile by classifying discovered attributes through using our defined decision trees and FMFs. Sequence of classified features per each profile is sent to a defined behavior HMM model to extract semantics, i.e., associated concepts, and to learn normal/abnormal behavior of services and attacks. The Accuracy of SM prediction for profiles' behavior is evaluated every reasoning time period, i.e., 100 seconds. In our HMM model, we defined five concepts, namely, known attack, UDP flood attack, TCP_Syn_flood_attack, normal and abnormal flow. Those

concepts depend on extracted and classified features per each profile, which are related to number of each profile in memory, packet size, port number, number of synchronization packets. Normal/abnormal behaviors for services are defined in HMM model based on values' range of input features. For instance, a normal behavior of TCP services is assigned by certain range of port numbers and packet sizes. We propose four cases of operation. The first case is that NetMem operation before learning that SM does not learn and derive concepts yet. The second one is operation after learning concepts; and SM will perform concept matching processes for learned data patterns. The third case is operation before learning and with DVA control. DVA control is a simple mechanism proposed to minimize storage space in memory that each data profile will not exceed a ratio of total number of profiles in memory. The last case is operation after learning and with DVA control.

Figure 6.13 shows measured average throughput at the two legitimate TCP/UDP destinations in cases of operation with/without NetMem before/after learning and with/without DVA control. Figure 6.13 (a, b) shows operation without NetMem that according to abnormal TCP/UDP flows, QoS of TCP service is deteriorated and also much resources consumption at UDP destination due to UDP flood attacks initiated from non-legitimate entities. In Figure 6.13 (c, d), the operation is with NetMem, however,



(a) Throughput at TCP destination without NetMem

(b) Throughput at UDP destination without NetMem

(c)Throughput at TCP destination with NetMem before learning

(d) Throughput at UDP destination with NetMem before learning

(e)Throughput at TCP destination with NetMem after learning

(f) Throughput at UDP destination with NetMem after learning

Figure 6.13 Throughput at TCP/UDP Destinations with/without NetMem and before/after learning

memory does not maintain concepts that can be matched with learned patterns of running services' profiles. So, SM begins to learn and derive concepts during the simulation time. SM learns patterns of three service profiles in memory which reveal two types of attacks and an abnormal flow behavior. Accordingly, three concepts are constructed and stored. Routers in our scenario learned those concepts through linkage between known IPs for those routers and service types attributes in profiles in memory, and extracted semantics and thereby routers stopped allocating resources and processing data concerning abnormal services' flow. Hence, QoS of legitimate services begins to enhance, however, after overhead time due to the patterns learning phase by SM, and memory access and semantics retrieval by routers. Legitimate UDP flow stops because arrival of many datagr s from attackers increases probability of collision which make buffer overflow and result in exceeding legitimate flow for the client and reply timeout value (30 seconds). In Figure 6.13 (e, f), NetMem operates after learning patterns and maintaining semantics for abnormal flows and attacks. So, routers learn early semantics of their services and they limit resources assigned to abnormal flows.

SM keeps learning data patterns in the storage memory and it matches them with registered semantics. Figure 6.14 illustrates SM timeliness to analyze and learn patterns of service profiles, which are updated continuously at runtime, to derive semantics accordingly and to predict services' behavior. In cases operation after learning, SM takes time longer than cases of operation before learning because SM learned semantics of attacks and abnormal profiles and it found matching between analyzed and learned patterns and those semantics. Accordingly, routers will not assign resources for those profiles and normal traffics of TCP/UDP services will reach destinations successfully (data collisions and channel access contention are minimized). So, network throughput will be enhances and NetMem will be able to collect large number of normal profiles. Hence, size of the storage memory increases after learning as shown in Figure 6.15. NetMem achieved the same rate of learning in case of operation before learning

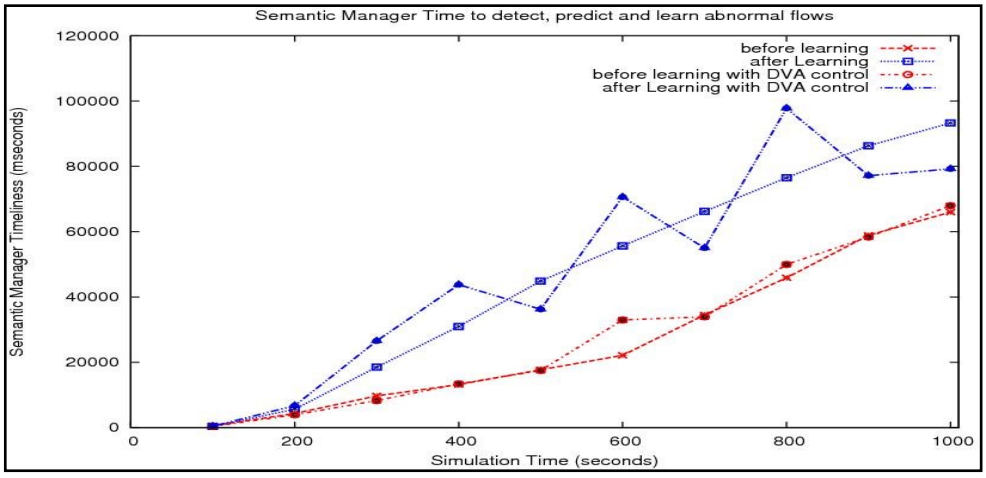


Figure 6.14 Semantic manager timeliness

with/without using the simple DVA control. In addition, NetMem, with DVA control after learning, succeeded in minimizing the required storage space in the memory. However, SM discovered 100% of semantics.

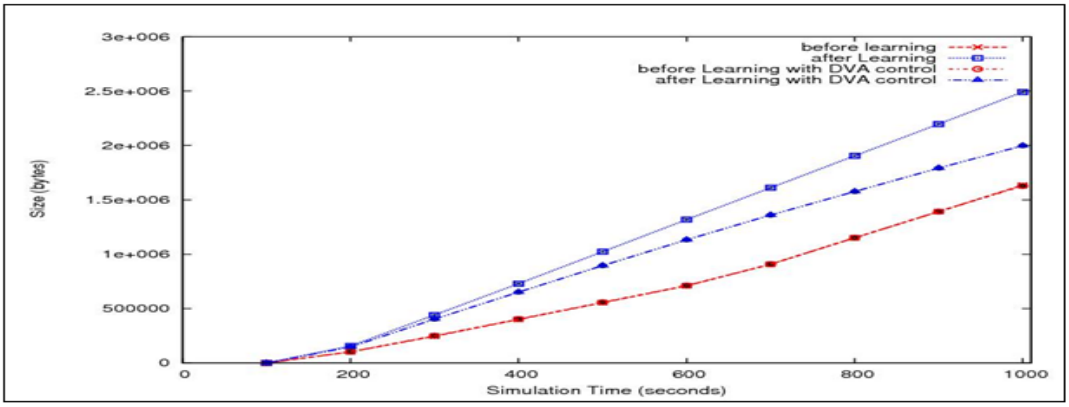


Figure 6.15 Data scalability in the storage memory with/without DVA Control

Conclusion

Learning and extracting network-semantics would help in a) better identification and prediction of anomalies and emergent behavior; and b) supporting more efficient decision making capability.

6.4.1.2 Small-scale Scenario II

We implemented a simple scenario for a NetMem-based network composed of four hosts and four routers as shown in Figure 6.16. There are two different service classes for data transfer where the first service class uses TCP transport protocol between two hosts and the other service class uses file transfer protocols on top of UDP between the other two hosts. We build a static route for each service class where intermediate routers transfer data packets of both services.

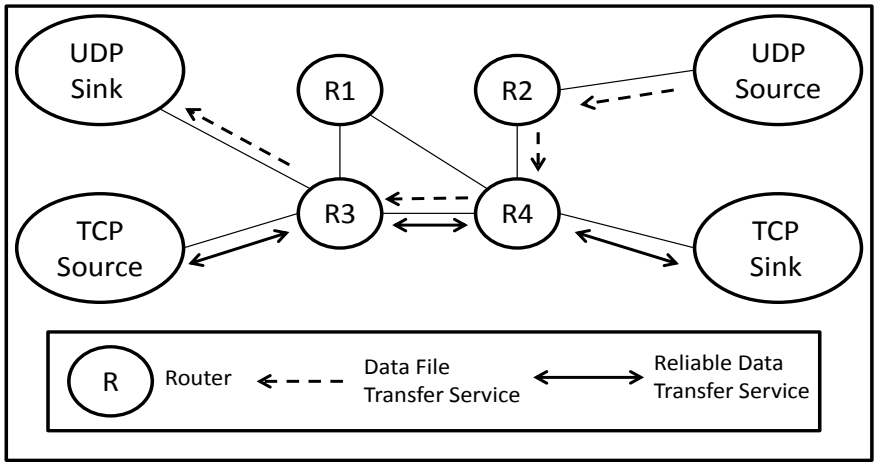


Figure 6.16 Simulated network layout

Table 6.4 shows the simulation parameters. Communicating entities in the scenario access NetMem at runtime and on-demand to store/discover/retrieve data and to learn semantics concerning their interesting services. For instance, the TCP and UDP services are provided in a specific region and at certain periods of time through the day. Our assumption here is that data regarding TCP and UDP services are collected via DVA from various Internet elements (i.e., sources, sinks and routers) and these data are uniformly represented in the storage memory as profiles, which show for example source/destination IP, service type, packet size, allocated buffer size and bandwidth, and service duration. These profiles exhibit patterns, which can be learned by the SM. Accordingly, the SM derives, using simple statistical-analysis-based reasoning model, semantics related to TCP and UDP services. Learned data patterns and extracted semantics enable other entities, such as the bottleneck router (i.e. router R3 in our scenario), to know QoS requirements of TCP/UDP-based services and occurrence time of running services.

For example, NetMem has semantics for the TCP service which reveal that this service, within range of IPs, uses specific resources at certain time period and at particular area because of the impact of another service (i.e., UDP service). Data (i.e., data profiles) and semantics in the storage memory are updated continually at runtime. In this scenario, we assume that there are 20 data profiles maintained in the storage memory where each service has 10 profiles. We show the capability of the bottleneck router to learn semantics of TCP service derived in NetMem to enhance the service's QoS by allocating required resources and to aid in detecting abnormal TCP flows.

Table 6.4 Simulation parameters of the third small scale NetMem-based network simulation sceanrio

Parameter	***Value***
Link Data Rate (fixed)	1 Mbps
Bottleneck Link Data Rate (variable)	10 Kbps
Bottleneck Router 3 Buffer Size (variable)	6000 packets
TCP MSS (fixed)	512 bytes
TCP MCWS (fixed)	128 bytes
TCP Time to Live (fixed)	255 seconds
UDP Packet Size (fixed)	700 bytes
Propagation Delay (fixed)	100 mseconds
JPG Photo size (fixed)	58 Kbytes
Simulation Time	1000 seconds

SM uses a simple associative rule learning algorithm with Apriori algorithms, for learning attributes of data profiles stored in the storage memory. SM calculates a weight

for each analyzed profile based on classified attribute' values of each profile. For profiles classification, SM adopts fuzzy set theory (fuzzy membership functions) where extracted profiles' attributes could be fuzzy and yield multi-valued classes for those attributes which accordingly refer to different semantics. We develop trapezoidal membership functions for three target attributes (bandwidth, buffer size and service duration) for data profiles in the storage memory. Each membership function outputs three classes with various degrees based on numerical values of these attributes compared to defined functions' split points. There are 20 service profiles in the memory as training sets where there are 10 profiles for the TCP service and the other 10 profiles are for UDP service. Based on extracted valued attributes, calculated data profiles weights D_w, and results from the used statistical model (e.g., number of profile occurrence (P_n) and interesting attributes (C_p) per each profile) besides sets of defined rules (e.g., if P_n>threshold_1 & C_p ==3 & D_w> threshold_2 then define service semantics based on classified attributes), SM can recognize patterns for TCP and UDP services and it develops their semantics.

Figure 6.17 and Figure 6.18 show measured throughput at TCP and UDP destinations in cases of operation without and with NetMem, respectively. These throughput data besides other data regarding for example operational parameters and protocols used for both services and required resources are maintained in data profiles in the storage memory.

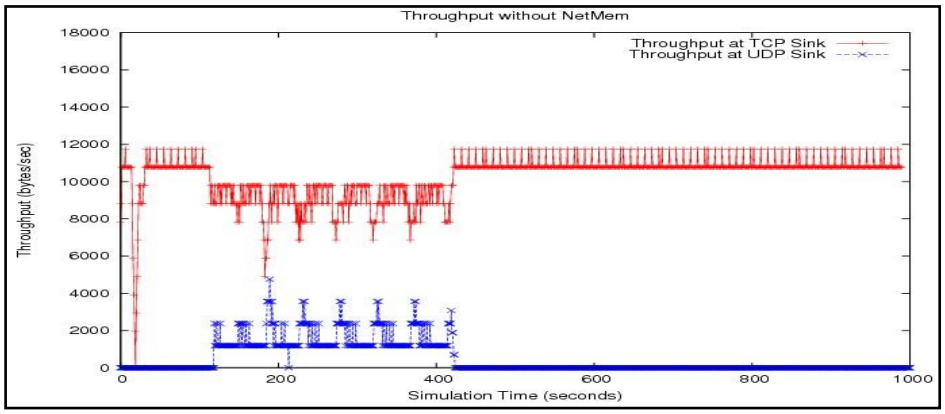


Figure 6.17 Throughput at destinations without NetMem

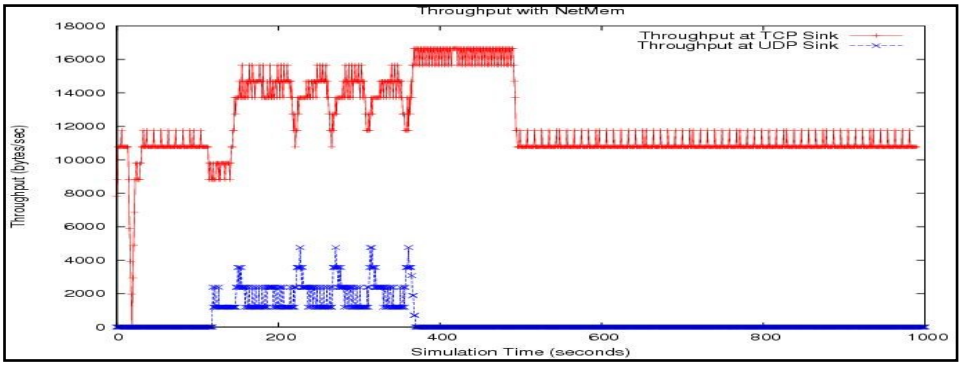


Figure 6.18 Throughput at destinations with NetMem

SM monitors and learns at runtime patterns related to these data profiles. From learned patterns, it knows that a UDP service starts at certain time and lasts for a period and the TCP service requires specific resources (e.g. buffer size and bandwidth) in that period to meet QoS levels (e.g. minimum throughput level at TCP destination).

Figure 6.19 shows throughput at the bottleneck router for TCP flow with/without NetMem. In operation with NetMem, the router adapted to the effect of UDP flow on the TCP service's QoS based on learned semantics in memory. Based on recognizing TCP and UDP semantics, we notice that the router was able to self-configure its system to optimize TCP service. Semantic inform the router to allocate larger buffer size and higher bandwidth for the TCP flow using the time period from 120 to 500 through the simulation time. Figure 6.20 shows the case that the router detected abnormal pattern of a TCP traffic generated in another simulation run. Router learned traffic abnormality via SM which matches learned pattern with already known semantics for TCP service maintained in the memory. Based on matching results, NCI component sent an attack alert to router. Accordingly, the router constrained that TCP service and enhanced the UDP service using

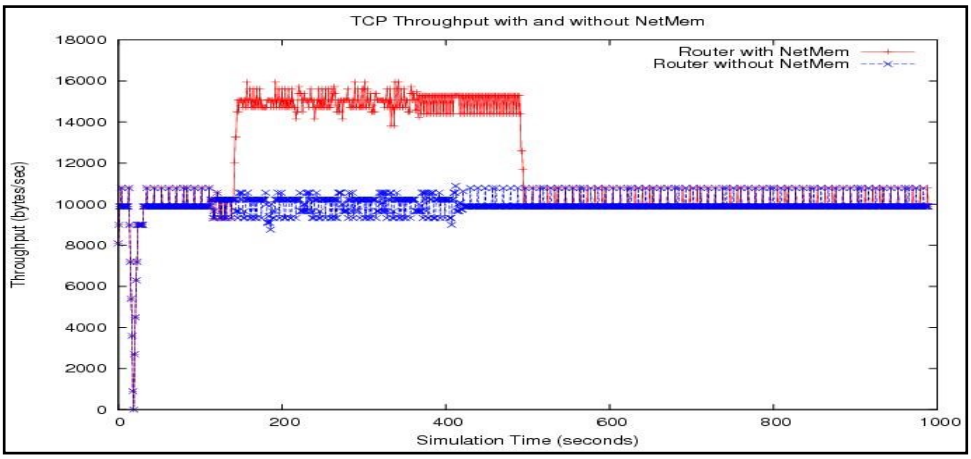


Figure 6.19 TCP Throughput at bottleneck router with/without NetMem

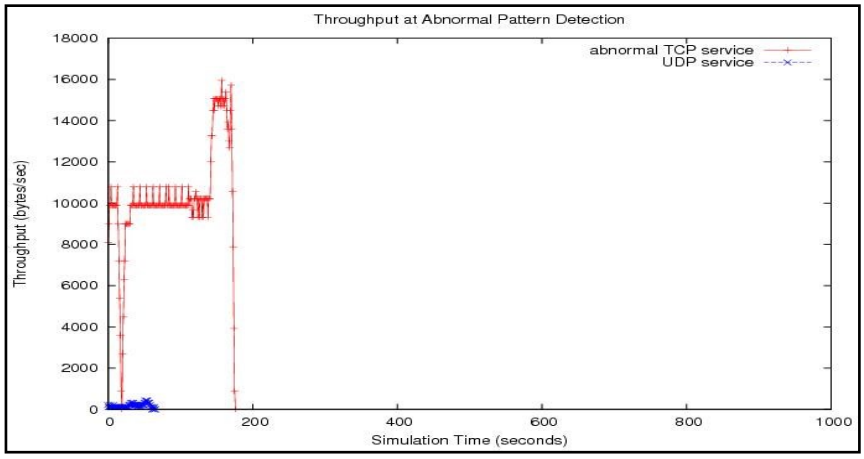


Figure 6.20 Throughput at bottleneck router with abnormal TCP pattern

its available resources.

Conclusion

- Networking entities (e.g., routers) along a communication path can enhance, at runtime, services' QoS by accessing and learning semantics of those services.
- Learning and extracting network-semantics would help in a) better identification and prediction for anomalies and emergent behavior; and b) supporting for an efficient decision making capability.
- Better utilization of resources and improvement in situational awareness capabilities can be achieved at runtime with NetMem by recognizing changes in patterns (e.g., QoS requirements) of services.
- Security can be enhanced by recognizing and differentiating normal and abnormal data patterns in NetMem.

6.4.1.3 Large-scale NetMem-based network simulation scenario using real offline datasets

In this scenario, we study the effectiveness of NetMem semantics reasoning operations using various reasoning models and adopting real offline datasets for learning patterns and extracting semantics of normal/abnormal flows of TCP- and UDP-based services and related attacks (e.g., TCP-SYN flood attacks). Extracted semantics are represented as correlated concept classes in NetMem. Networking entities can access and learn those concept classes at runtime and on-demand to enhance QoS of their operations.

NetMem is designed as one NMemAgent. The simulation scenario is run at adopting various semantics reasoning models in NemMem. The utilized semantics reasoning models are: LDA-based model, HMM-based model and the hybrid model (or the LDA-HMM model). We study the impact of using NetMem on networking operations performance when NetMem employs different reasoning models. We compare obtained results at using NetMem with the results at operation without using NetMem. Table 6.5 shows the addressed metrics for analyzing performance of the semantics reasoning process executed by NetMem.

Table 6.5 Performance analysis metrics of semantics reasoning techniques

Metric	*Function*	*Equation*	*Unit*
Average Throughput (Thp)	Describes data throughput at a current simulation time by the summation of current captured data packets' sizes (S_p) in a defined window divided by the window size (5 seconds)	$Thp = \sum (S_p)/(5$ seconds window size).	bytes/sec
Network Latency (L)	Measures the consumed time from the beginning of simulation until learning all behavior classes of running	$L = T_R + T_S$ where T_R is the time-overhead caused by the reasoning technique and T_S is the time period	mseconds

	file transfer services	measured from simulation start	
Packets Delivery Ratio (P_D)	Calculates ratio of successful packets received (P_R) by network destination related to number of sent packets (P_S)	$P_D = P_R / P_S$	--
Prediction Accuracy (A_c)	Calculates ratio of true positive (Tp) classes and true negative (Tn) classes that are learned with respect to all behavior classes (N_c) that should be learned	$A_c = (Tp+Tn) / N_c$	--
False Positive (Fp) Ratio	Calculates ratio of normal behavior classes that misclassified to abnormal classes according to N_c	Fp / N_c	--
False Negative (Fn) Ratio	Calculates ratio of abnormal behavior classes that misclassified to normal classes according to N_c	Fn / N_c	--
Recall (R)	Measures the effectiveness of system to learn abnormal behavior classes (including classes of attacks) with respect to Fn	$R= Tp/(Fn+Tp)$	--

Our simulator is based on J-Sim [131, 132]. We adapted already existing java codes of HMM and LDA [139, 140]. We implemented a code for the LDA-HMM model and integrated it with J-Sim. Random fully-connected static network topologies were generated for 50 nodes using minimum degree proximity algorithm [141]. TCP- and UDP-based file transfer services were run among nodes. Some nodes were chosen to send malicious and/or abnormal data (e.g., attacks like TCP SYN-flood attack and UDP flood attack). Exchanged data among nodes are represented and maintained in NetMem as profiles of attribute-value pairs. Those attributes give information about, for example, service type, packet type and packet size. KDD'99 dataset with 41 data attributes [133] was used as offline dataset by NetMem to learn semantics of normal TCP flows and some DoS attacks, such as TCP SYN-flood attack. DVA registers data in NetMem with/without using LSH. DVA applies an LSH algorithm to reduce dimensionality of stored data. SM learns patterns of registered data, recognizes their group of attributes and classifies those attributes adopting our implemented codes for ARL and FMF. SM utilizes a simple statistical-analysis-based model to get statistics about kept profiles in NetMem (e.g., knowing profiles which have same attributes with similar classification). SM extracts and classifies high-level latent features using the implemented semantics reasoning model. We tested three different semantics reasoning models which are 1)

LDA-based model; 2) HMM-based model; and 3) the LDA-HMM or hybrid model. We studied the capability of these models to learn data semantics and to know various concept classes related to normal/abnormal flows and attacks.

NetMem executes behavior classification for analyzed flows of TCP- and UDP-based services. Extracted semantics concerning those flows will form ontology of concept classes which show for example classes for the normal_TCP-based_service, normal_UDP-based_service, attack_TCP-based_service and attack_UDP-based_service. An effective semantics reasoning model is the model which has the ability to discover all or most classes of concepts related to running services in the network. NetMem generates alerts when it detects matching between what it learns at real-time and maintained concept classes. A performance analysis was done for the operation of semantics reasoning process by SM. Each implemented semantics reasoning model was evaluated based on the detected true positives (Tp) and negatives (Tn) besides false positives (Fp) and negatives (Fn). We also showed the recall ratio of each model where that ratio presents the effectiveness of the model in learning classes of attacks accurately. Table 6.6 shows simulation parameters and their default values.

Table 6.6 Simulation parameters of the large-scale NetMem-based network simulation scenario

Parameter	***Default value***	***Unit***
Number of nodes	50	node
Number of attributes/KDD data set	41	attribute
Link data rate	1	Mbps
Router buffer size	7000	packet
TCP MSS (normal/abnormal)	512/1024,2048	byte
TCP MCWS	128	byte
TCP Time to Live	255	seconds
UDP packet size (normal/ abnormal)	512/512,2048	byte
UDP Client/Reply Timeout	30	seconds
Propagation model	Free space model	--
Propagation delay	100	mseconds
MAC protocol	IEEE 802.11	--
Routing protocol	AODV	--
Rate of NetMem access by hosts	1	time/11 seconds
Rate of patterns learning by SM	1	time/10 seconds
HMM approach/number of training sequences	Unsupervised with Baum-Welch algorithm/1000	--
LSH (L hash tables, K hash function length)	(L=6,K=3)	--
LDA (# of iterations)	20000	--
Simulation Time	100	seconds

We repeated the experiment nine times for each implemented semantics reasoning model (LDA, HMM or LDA-HMM) in NetMem and also when operation at different cases

which are: a) without using LSH; b) using undirected LSH and c) using directed LSH. Directed LSH mechanism means that the implemented LSH algorithm adopts a specific attributes' group smaller than the one used by the undirected LSH and related to interesting file transfer services. Semantics reasoning models were designed to reason about data semantics clarifying the normal and abnormal behavior of TCP- and UDP-based service flows. Recognized semantics of abnormal service flows' behavior might refer to known attacks (e.g., TCP-SYN-flood attack and UDP-flood attack). We have 13 behavior concept classes which have to be learned. Five classes represent normal behavior and nine classes define abnormal behavior. The effectiveness of NetMem and its impact over networking operations was studied via measuring average network throughput, network latency and percentage of successful packets received at network destination. Figure 6.21 illustrates average network throughput measured at network destination when using NetMem with/without LSH. The LDA-HMM model learned most data semantics (i.e., behavior concept classes) of normal/abnormal flows earlier, besides clarifying attacks in the network. This enabled NetMem to generate early alerts that allowed nodes to recognize semantics and to subsequently suppress abnormal flows and attacks.

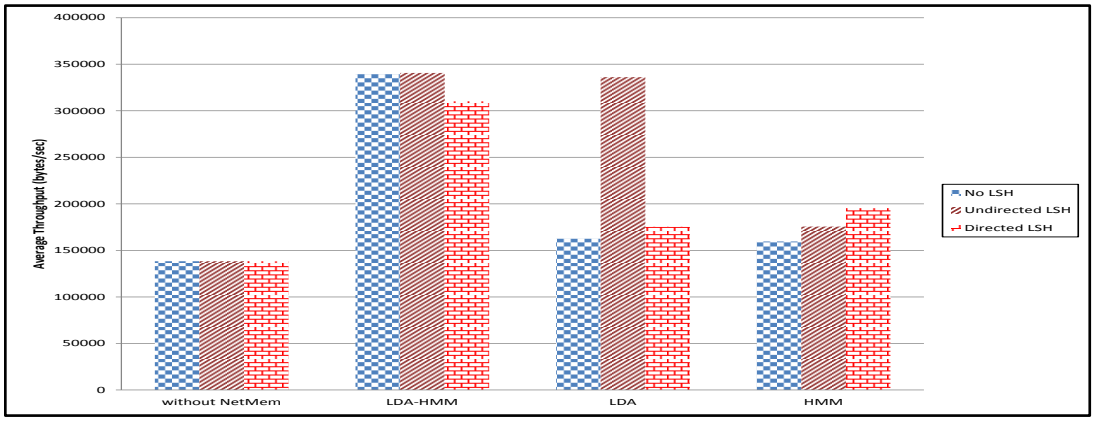
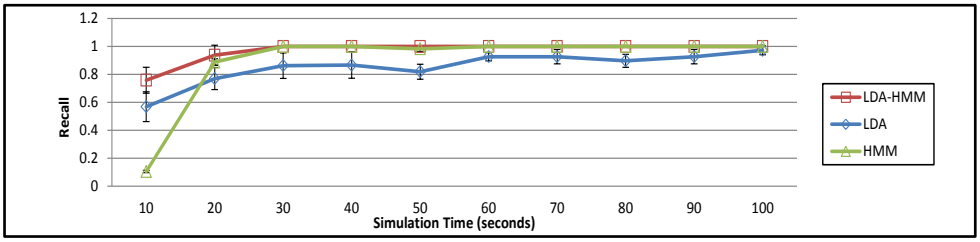
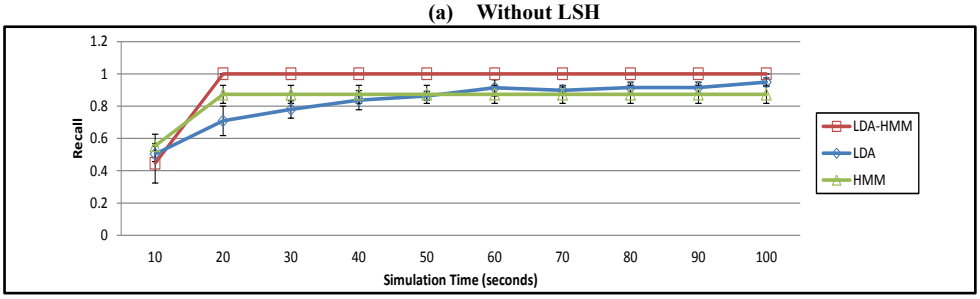


Figure 6.21 Average network throughput

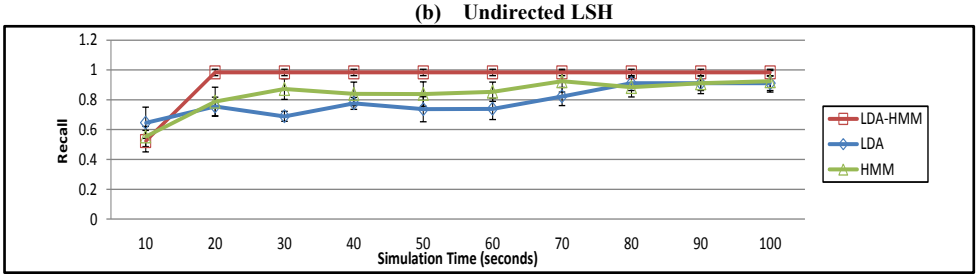
Figure 6.22 shows NetMem system recall at using different semantics reasoning models. In Figure 6.23, The LDA-HMM model was able to learn semantics with higher level of accuracy through the simulation time compared with LDA and HMM. This showed the ability of the HIT to mitigate challenges faced by the HMM to work with reduced-dimensional data. It can be concluded from obtained results that NetMem operations with the hybrid reasoning model achieved low false negative ratios compared with NetMem operations at adopting monolithic intelligence-based models. The usage of LSH saved storage space at NetMem. Figure 6.24 and Figure 6.25 show Fn and Fp ratios of the NetMem system with different semantics reasoning models. The hybrid model for semantics reasoning achieved better values compared with other models.



(a) Without LSH

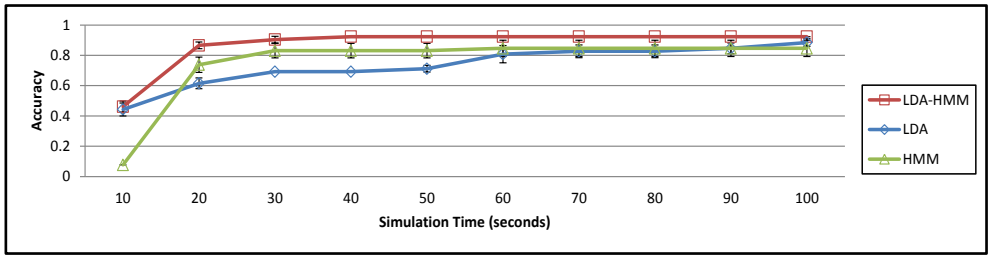


(b) Undirected LSH

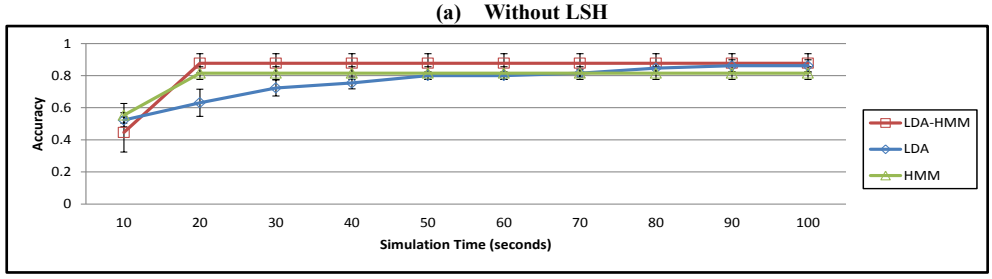


(c) Directed LSH

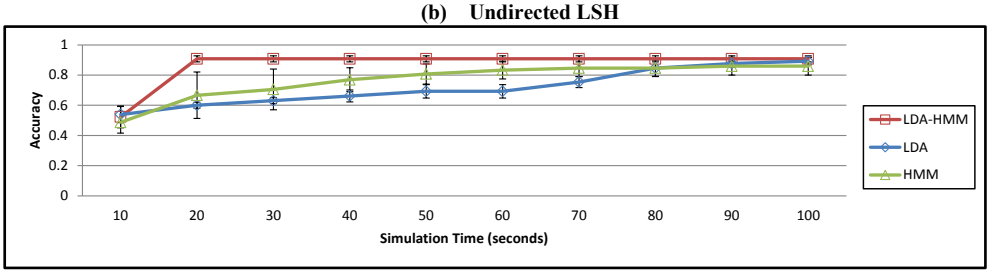
Figure 6.22 NetMem system recall for learning semantics



(a) Without LSH

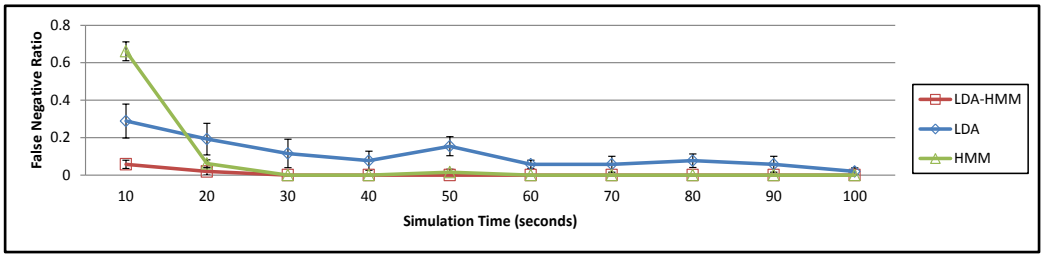


(b) Undirected LSH

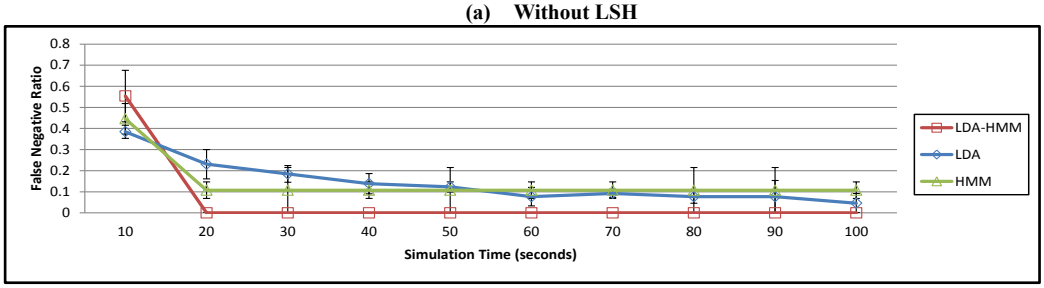


(c) Directed LSH

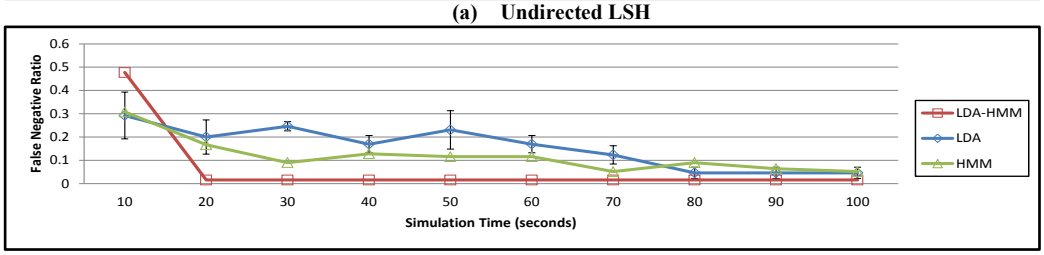
Figure 6.23 NetMem system accuracy for learning semantics



(a) Without LSH

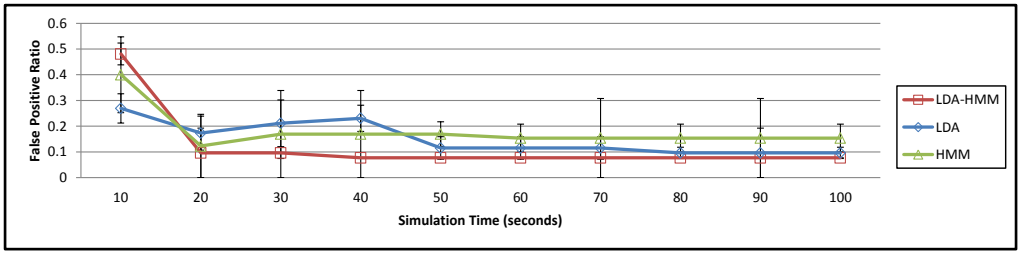


(a) Undirected LSH

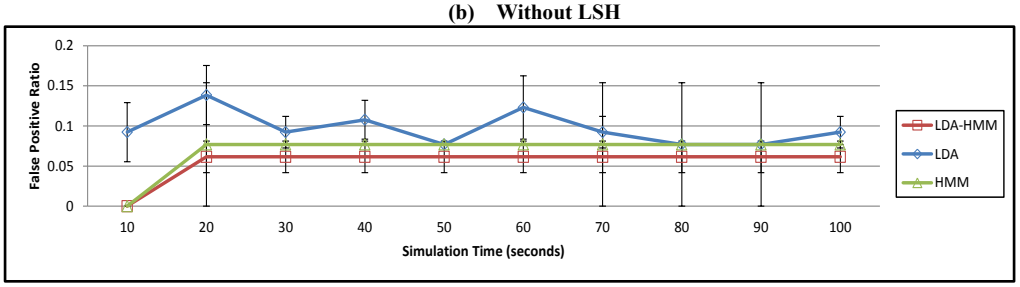


(a) Directed LSH

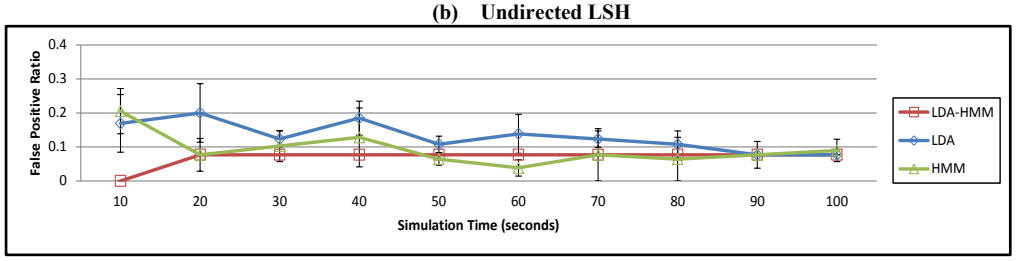
Figure 6.24 NetMem system false negative (Fn) ratio for learning semantics



(b) Without LSH



(b) Undirected LSH



(b) Directed LSH

Figure 6.25 NetMem system false positive (Fp) ratio for learning semantics

Figure 6.26 depicts the percentage of packets received successfully by the destination. The hybrid model achieved the best packet delivery ratio. That model succeeded in recognizing early most behavior concept classes of normal/abnormal flows of running services, which resulted in having higher network throughput.

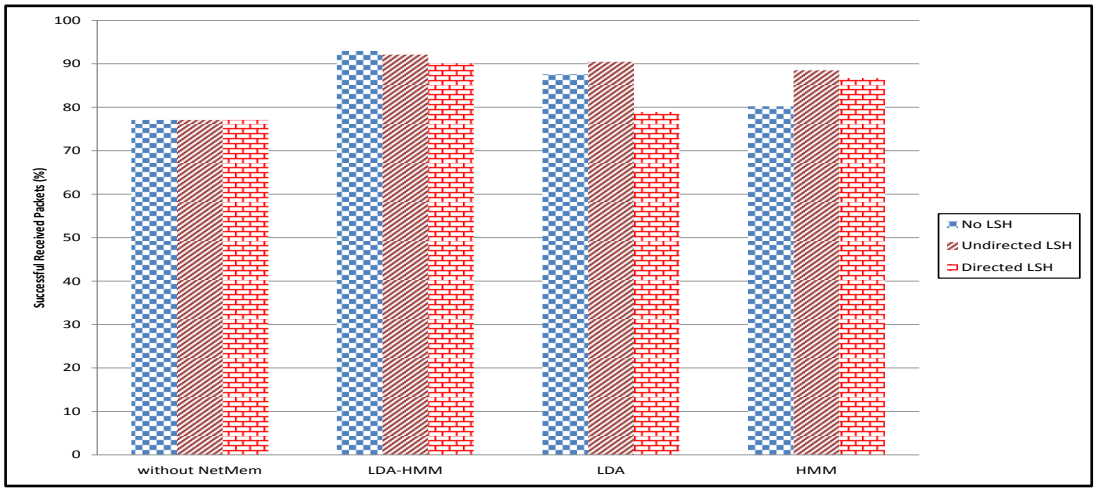


Figure 6.26 Percentage of successful packets received at network destination

Figure 6.27 illustrates the effectiveness of NetMem to learn, in a timely manner, semantics of normal/abnormal flows and to detect running malicious flows and attacks accordingly. Due to the time-overhead caused by semantics reasoning models in NetMem to learn patterns and extract semantics, there was time delay experienced in the network to accomplish NetMem tasks. Figure 6.28 shows the capability of directed or undirected LSH to reduce space needed in the storage memory. In spite of space reduction, NetMem can still effectively learn semantics of abnormal flows and attacks.

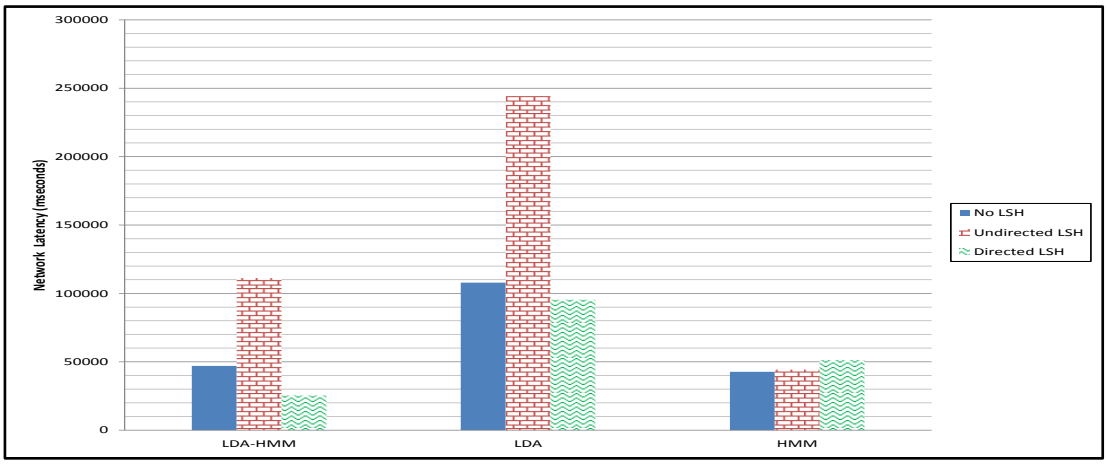


Figure 6.27 Network latency due to the usage of NetMem

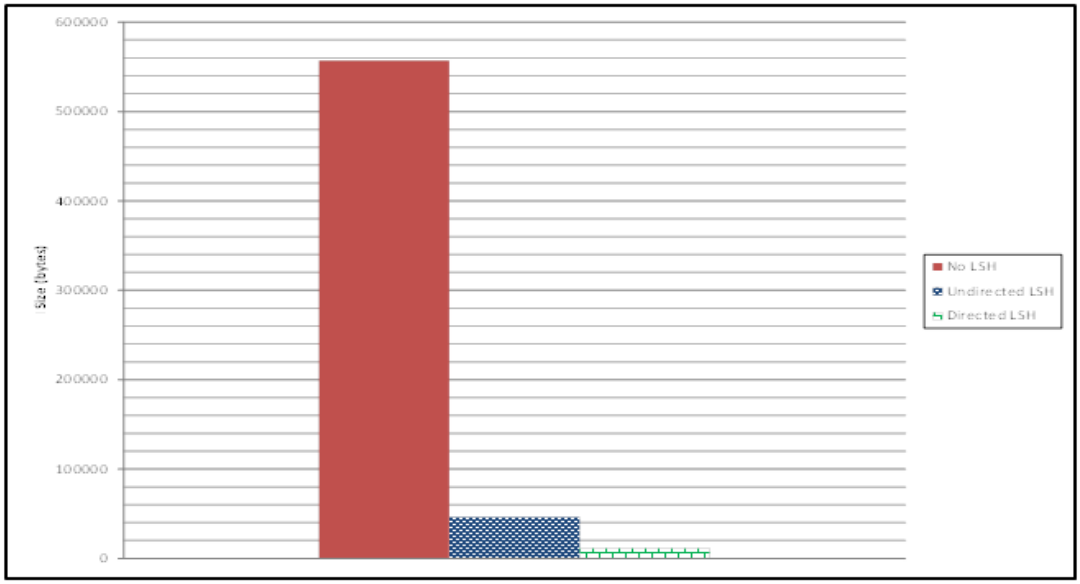


Figure 6.28 Storage space saving in NetMem storage memory due to the usage of LSH

Conclusion

1- The LDA-HMM-based model for semantics reasoning has overcome limitations of other models with individual intelligence techniques (LDA or HMM). Consequently, better performance (higher accuracy and recall with low Fp and Fn) of semantics reasoning process was achieved in case of using the LDA-HMM model; and

2- The LDA-HMM model was able to learn features in case of having full- or reduced-dimensional data profiles and reason about network-semantics. This led to recognize, in case of using or not using LSH, almost all behavior classes which are related to different Internet elements (normal file transfer services and attacks). Consequently, this aided in enhancing QoS of running services.

6.5 Conclusion

The effectiveness of NetMem and its implemented reasoning models was evaluated via simulation through implementing a) small-scale NetMem prototype over real Internet traffic data; and b) simulation scenarios using J-Sim. Simulation results showed the capability of NetMem to learn patterns of large-scale data with full or reduced dimensionality and reason about semantics that are used in a) enhancing QoS of running services; b) detection of anomalies and attacks; and c) learning normal/abnormal behavior classes of some Internet elements (e.g., services and attacks). Also, simulation results showed the capability of NetMem to support existing networking tools (e.g., intrusion detection systems, such as snort) to execute their functions more effectively and efficiently. The LDA-HMM-based reasoning model achieved higher NetMem effectiveness whether using full- or reduced data dimensionality.

Chapter 7

7 Conclusion and Future Work

7.1 Conclusion

In this dissertation, we presented a biologically-inspired customizable application-agnostic distributed network memory management system, termed NetMem, for addressing the "Internet Semantics Gap".

NetMem is a distributed multi-agent system that mimics functionalities of the human memory in learning patterns of high- and multi-dimensional data and reasoning about associative network-semantics in order to develop dynamic network-concept ontology and to have semantic-driven network operations. From a system's perspective, NetMem can be viewed as an overlay network of distributed "memory" agents located at multiple levels targeting different levels of data abstraction and scalable operations. Figure 7.1 shows the main operations of NetMem:

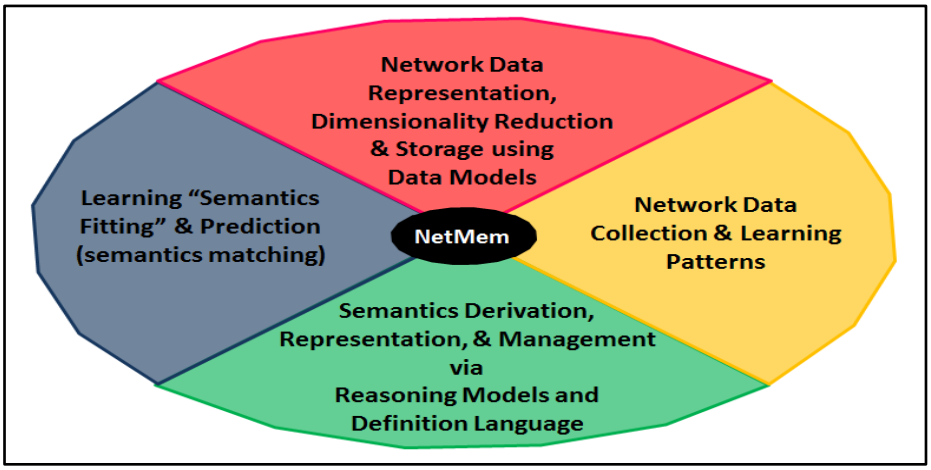


Figure 7.1 NetMem operations

The main contributions of our work are outlined as follows.

Intellectual Merit

- Biologically-inspired, customizable, application-agnostic distributed network memory management system with efficient processes for extraction of classified high-level features and reasoning about rich semantics in order to resolve the Internet semantics gap and target Internet intelligence.
    - Distributed system for efficient learning of patterns of multi-dimensional data, with full or reduced dimensionality, and semantics with lower levels

of abstraction and reasoning about semantics with higher levels of abstraction through providing:

- Multi-level system with a set of agents for each level in order to support scalable semantics learning and management operations.
- Memory system structure comprising:
  - Short-time Memory (StM): It maintains for short-time raw data and semantics with lower levels of abstraction, enabling learning patterns and extracting semantics with higher levels of abstraction;
  - Long-term Memory (LtM): It maintains for long-time less dynamic semantics with higher levels of abstraction, and offers accessible correlated concept classes related to various network concerns for matching and prediction processes.

- Semantics management methodology using monolithic and hybrid intelligence techniques for managing efficiently data semantics and building runtime-accessible dynamic ontology of correlated concept classes related to various Internet elements and at different levels of abstraction and granularity.
- Analytical model and study for quantitatively evaluating NetMem scalability and response time in the case of retrieving data, and processing time in the case of data abstraction.

Broader Impact

- Support for semantics-driven and semantics-aware operations to enable "smarter" networking as follows.
  - Internet elements and tools will better understand traffic and the behavior of elements in real-time, on-demand, and at low cost (providing or strengthening self-awareness, adaptation, and evolution capabilities).
  - Resolving the Internet semantic gap between network activities and the decision making process for better understanding of network dynamics.
  - Supporting existing networking tools (e.g., intrusion detection systems) to execute their functions more efficiently and effectively (e.g., enhance the prediction accuracy of tools).
- Support for autonomic network-semantics discovery and responses via accessible and shared network concept ontology enhancing or providing:
  - Anomaly and emerging behavior discovery;
  - Efficient decision making capability due to semantics matching processes at different levels of granularity;
  - Dynamic QoS provisioning;

  - Resource utilization (e.g., in the case of pre-allocating resources for unfamiliar services);
  - Predictive operation (e.g., early detection and prediction of attacks)..
- Providing dynamic network concept ontology related to various Internet elements and at different levels of abstraction showing FBS aspects of maintained concepts
  - Advancing network science and engineering to include evolution, modeling and simulation studies.

The NetMem evaluation studies via simulation using real Internet traffic data presented in this dissertation illustrated the efficacy and success of NetMem in learning and utilizing data semantics for anomaly detection and enhancing QoS of running services. The studies demonstrated NetMem's benefits and performance using different reasoning models with/without adopting data dimensionality algorithms over low and high volume of real traffic data to learn the behavior of normal and anomalous flows and attacks. Additionally, our analytical study evaluating NetMem as a multi-agent system showed good levels of NetMem response time and efficiency for fulfilling information requests by adopting various semantics reasoning models over full or reduced data dimensionality. Optimized configurations for the NetMem system (via implementing NetMem agents with specific reasoning models obtained by means of solutions of a formalized optimization problem) enhanced the overall NetMem system efficiency.

NetMem Limitations

- Data semantics reasoning, discovery and retrieval from NetMem might result in some time overhead (i.e., NetMem response time) which affects network latency
- Implemented semantics reasoning algorithms and constructed behavior extraction models might affect the accuracy and the level of abstraction of discovered concepts.

7.2 Future Work

Our future work includes the following:

1- Designing more efficient adaptive mechanisms for adjusting and configuring autonomously the NetMem system operation parameters, such as the reasoning window size (i.e., the frequency of executing reasoning processes) of implemented semantics reasoning models. This will help in improving the efficiency and effectiveness of NetMem. Some criteria can be mixed with different weights, for example, the total size of raw data in the storage memory and the change rate in the amount of data related to specific concerns in the memory.

2- Designing and evaluating qualitatively and quantitatively the operation performance of specific network concept class-oriented NetMem agents (NetMem levels with specialized agents which reason about semantics of certain network concerns or features or dimension). Also, the impact of having NetMem agents in the same or

different NetMem levels with conflict of interests on NetMem efficiency (throughput and response time) will be studied.

3- Full simulation of a NetMem-enabled network considering the hierarchical structure of NetMem with distributed autonomous NetMem agents. This simulation will aid in realizing more optimality for NetMem configuration with heterogeneous intelligent agents implementing various semantics reasoning models.
4- Prototyping NetMem over a large-scale real network with different operation conditions clarifying the interaction amongst multiple NetMem agents and the construction of network-concept ontologies depending on various formation criteria (e.g., specialized ontologies related to communication protocols or file transfer services).
5- Investigating real-life systems, such as smart electric power grids, and simulating NetMem capabilities for them to help in improving the performance of the system operations and jobs. NetMem can strengthen the security of state estimation processes in smart grids by providing dynamic models for state estimation which clarify the normal/abnormal behavior of multiple smart grid elements, like remote terminal units, power generation units, and network entities (e.g., routers) in communication networks of smart girds.

Publications

Published Papers

1- B. Mokhtar and M. Eltoweissy, "Memory-enabled autonomic resilient networking," in *Collaborative Computing: Networking, Applications and Worksharing (CollaborateCom), 2011 7th International Conference on*, 2011, pp. 132-141.
2- B. Mokhtar and M. Eltoweissy, "Biologically-inspired network "memory" for smarter networking," in *Collaborative Computing: Networking, Applications and Worksharing (CollaborateCom), 2012 8th International Conference on*, 2012, pp. 583-590.
3- B. Mokhtar, M. Eltoweissy, and H. Elsayed, "Network memory system for enhanced network services," in *9th International Conference on Innovations in Information Technology,* March 17-19, 2013. Al Ain, UAE.

Accepted Papers

1- B. Mokhtar and M. Eltoweissy, "Towards a Data Semantics Management System for Internet Traffic", *the 6th IEEE-IFIP International Conference on New Technologies, Mobility and Security (NTMS).*
2- B. Mokhtar and M. Eltoweissy, "Hybrid Intelligence for Semantics-enhanced Networking Operations", *the 27th International Conference of the Florida Artificial Intelligence Research Society (FLAIRS-27).*

Submitted Papers

1- B. Mokhtar and M. Eltoweissy, Semantics Management System for Big Networks, submitted to *the ElSEVIER Journal of Network and Computer Applications (JNCA).*

2- B. Mokhtar and M. Eltoweissy, Biologically-inspired Network "Memory" Management with Hybrid Intelligence, submitted to *the Wiley international journal of intelligent systems.*

References

[1] A. Feldmann, "Internet clean-slate design: what and why?," *ACM SIGCOMM Computer Communication Review,* vol. 37, pp. 59-64, 2007.

[2] G. Bouabene*, et al.*, "The autonomic network architecture (ANA)," *Selected Areas in Communications, IEEE Journal on,* vol. 28, pp. 4-14, 2010.

[3] A. Zafeiropoulos*, et al.*, "Monitoring within an autonomic network: a GANA based network monitoring framework," in *Service-Oriented Computing. ICSOC/ServiceWave 2009 Workshops*, 2010, pp. 303-313.

[4] J. Day*, et al.*, "Networking is IPC: A guiding principle to a better Internet," in *Proceedings of the 2008 ACM CoNEXT Conference*, 2008, p. 67.

[5] H. Hassan*, et al.*, "CellNet: a bottom-up approach to network design," in *3rd International Conference on New Technologies, Mobility and Security (NTMS)*, pp. 1-6, 2009

[6] *VSS Monitoring corporate*. Available: http://www.vssmonitoring.com/

[7] *Pervasive network intelligence and complete packet inspection, cPacket networks*. Available: http://www.cpacket.com

[8] D. Zhiming*, et al.*, "Massive Heterogeneous Sensor Data Management in the Internet of Things," in *Internet of Things (iThings/CPSCom), 2011 International Conference on and 4th International Conference on Cyber, Physical and Social Computing*, 2011, pp. 100-108.

[9] R. Khan*, et al.*, "Future Internet: The Internet of Things Architecture, Possible Applications and Key Challenges," in *10th International Conference on Frontiers of Information Technology (FIT)*, pp. 257-260, 2012.

[10] J. Srivastava*, et al.*, "Web usage mining: Discovery and applications of usage patterns from web data," *ACM SIGKDD Explorations Newsletter,* vol. 1, pp. 12-23, 2000.

[11] CAIDA. *Correlating heterogeneous measurement data to achieve system-level analysis of internet traffic trends*. Available: www.caida.org/projects/trends/ ,2002

[12] M. Allman*, et al.*, "A scalable system for sharing internet measurements," in *Proc. PAM*, 2002.

[13] *Apache hadoop*. Available: http://hadoop.apache.org

[14] D. Li*, et al.*, "Network Thinking and Network Intelligence," in *Web Intelligence Meets Brain Informatics*. vol. 4845, N. Zhong*, et al.*, Eds., ed: Springer Berlin Heidelberg, 2007, pp. 36-58.

[15] S. Mimaroglu and D. A. Simovici, "Approximate computation of object distances by locality-sensitive hashing," in *Proceedings of the 2008 International Conference on Data Mining, Washington, DC*, 2008.

[16] J. Hawkins and S. Blakeslee, *On Intelligence*: St. Martin's Press, 2005.

[17] W. Zeng*, et al.*, "Research on cloud storage architecture and key technologies," in *Proceedings of the 2nd International Conference on Interaction Sciences: Information Technology, Culture and Human*, 2009, pp. 1044-1048.

[18] K. McCloghrie and M. Rose, "RFC 1066-Management Information Base for Network Management of TCP/IP-based Internets," *TWG, August,* 1988.

[19] J. S. Gero, "Design prototypes: a knowledge representation schema for design," *AI magazine,* vol. 11, p. 26, 1990.
[20] K. Dorst and P. E. Vermaas, "John Gero's Function-Behaviour-Structure model of designing: a critical analysis," *Research in Engineering Design,* vol. 16, pp. 17-26, 2005.
[21] S. Paul, "An Optimized distributed association rule mining algorithm in parallel and distributed data mining with xml data for improved response time," *International Journal of Computer Science and Information Technology,* vol. 2, April 2010.
[22] M. Winter, *Goguen categories: a categorical approach to L-fuzzy relations*: Springer Publishing Company, Incorporated, 2007.
[23] D. M. Blei*, et al.*, "Latent dirichlet allocation," *the Journal of machine Learning research,* vol. 3, pp. 993-1022, 2003.
[24] L. Rabiner and B. Juang, "An introduction to hidden Markov models," *ASSP Magazine, IEEE,* vol. 3, pp. 4-16, 1986.
[25] S. Rozsnyai*, et al.*, "Large-scale distributed storage system for business provenance," in *IEEE International Conference on Cloud Computing (CLOUD)*, 2011, pp. 516-524.
[26] IBM. Available: http://www.ibm.com
[27] D. v. b. informatica. Available: http://www.informatica.com
[28] D. v. b. queplix. Available: http://www.queplix.com
[29] *Data virtualization by denodo technologies.* Available: http://www.denodo.com/en/solutions/technology/data_virtualization.php. 2009
[30] R. C. Atkinson and R. M. Shiffrin, "Human memory: A proposed system and its control processes," *The psychology of learning and motivation,* vol. 2, pp. 89-195, 1968.
[31] R. O. Duda*, et al.*, *Pattern classification*: John Wiley & Sons, 2012.
[32] I. Jolliffe, *Principal component analysis*: Wiley Online Library, 2005.
[33] D. D. Clark*, et al.*, "A knowledge plane for the internet," in *Proceedings of the 2003 conference on Applications, technologies, architectures, and protocols for computer communications*, 2003, pp. 3-10.
[34] D. F. Macedo*, et al.*, "MANKOP: A Knowledge Plane for wireless ad hoc networks," in *Network Operations and Management Symposium, 2008. NOMS 2008. IEEE*, 2008, pp. 706-709.
[35] A. Shieh*, et al.*, "NetQuery: a knowledge plane for reasoning about network properties," in *Proceedings of the ACM CoNEXT Student Workshop*, 2010, p. 23.
[36] P. A. A. Gutiérrez*, et al.*, "An advanced measurement meta-repository," in *Proceedings of 3rd International Workshop on Internet Performance, Simulation, Monitoring and Measurement*, 2005.
[37] D. H. Chau*, et al.*, "Apolo: making sense of large network data by combining rich user interaction and machine learning," in *Proceedings of the 2011 annual conference on human factors in computing systems*, 2011, pp. 167-176.
[38] I. Bíró and J. Szabó, "Latent dirichlet allocation for automatic document categorization," in *ECML PKDD '09 Proceedings of the European Conference on Machine Learning and Knowledge Discovery in Databases*, pp. 430-441, 2009.

[39] J. Jiten, *et al.*, "Semantic feature extraction with multidimensional hidden Markov model," in *Proceedings of SPIE*, 2006, pp. 211-221.
[40] T. Mori, *et al.*, "Typical behavior patterns extraction and anomaly detection algorithm based on accumulated home sensor data," in *Future Generation Communication and Networking (FGCN 2007)*, 2007, pp. 12-18.
[41] K. Yamanishi, *et al.*, "On-line unsupervised outlier detection using finite mixtures with discounting learning algorithms," in *Proceedings of the sixth ACM SIGKDD international conference on Knowledge discovery and data mining*, 2000, pp. 320-324.
[42] M. Nii, *et al.*, "Behavior extraction from multiple sensors information for human activity monitoring," in *IEEE International Conference on Systems, Man, and Cybernetics (SMC)*, pp. 1157-1161, 2011.
[43] E. Dodonov and R. F. de Mello, "A model for automatic on-line process behavior extraction, classification and prediction in heterogeneous distributed systems," in *Cluster Computing and the Grid, 2007. CCGRID 2007. Seventh IEEE International Symposium on*, 2007, pp. 899-904.
[44] L. Breiman, "Statistical modeling: The two cultures (with comments and a rejoinder by the author)," *Statistical Science,* vol. 16, pp. 199-231, 2001.
[45] M. Chen, *et al.*, "Semantic event extraction using neural network ensembles," in *Semantic Computing, 2007. ICSC 2007. International Conference on*, 2007, pp. 575-580.
[46] G. E. Hinton and R. R. Salakhutdinov, "Reducing the dimensionality of data with neural networks," *Science,* vol. 313, pp. 504-507, 2006.
[47] S. S. Cross, *et al.*, "Introduction to neural networks," *The Lancet,* vol. 346, pp. 1075-1079, 1995.
[48] T. K. Landauer, *et al.*, "An introduction to latent semantic analysis," *Discourse processes,* vol. 25, pp. 259-284, 1998.
[49] P. J. Van Laarhoven and E. H. Aarts, *Simulated annealing*: Springer, 1987.
[50] N. Cristianini and J. Shawe-Taylor, *An introduction to support vector machines and other kernel-based learning methods*: Cambridge university press, 2000.
[51] L. E. Baum, "An Inequality and Associated Maximization Technique in Statistical Estimation for Probabilistic Functions of a Markov Process," *Inequalities,* vol. 3, 1972.
[52] T. L. Griffiths, *et al.*, "Integrating topics and syntax," in *Advances in Neural Information Processing Systems*, 2004, pp. 537-544.
[53] L. Gilardoni, *et al.*, "LKMS–A Legal Knowledge Management System Exploiting Semantic Web Technologies," in *The Semantic Web–ISWC 2005*, ed: Springer, 2005, pp. 872-886.
[54] J. Zhou and R. Dieng, "A Semantic knowledge management system for knowledge-intensive manufacturing," in *2004 IADIS International Conference of e-Commerce, Lisbon, Portugal*, 2004.
[55] Z. C. Cob and R. Abdullah, "Ontology-based Semantic Web services framework for knowledge management system," in *Information Technology, 2008. ITSim 2008. International Symposium on*, 2008, pp. 1-8.
[56] J. Hendler, "Agents and the semantic web," *Intelligent Systems, IEEE,* vol. 16, pp. 30-37, 2001.

[57] J. Undercoffer, *et al.*, "Modeling computer attacks: An ontology for intrusion detection," in *RAID*, 2003, pp. 113-135.
[58] D. Willett and G. Rigoll, "Hybrid NN/HMM-based speech recognition with a discriminant neural feature extraction," *Advances in Neural Information Processing Systems,* pp. 763-772, 1998.
[59] R. Volner, "Human interaction system= intelligence network," in *11th International Biennial Baltic Electronics Conference*, 2008, pp. 217-220.
[60] M. Yabusaki, *et al.*, "Mobility management in All-IP mobile network: end-to-end intelligence or network intelligence?," *Communications Magazine, IEEE,* vol. 43, pp. supl. 16-supl. 24, 2005.
[61] C. C. Yang, *et al.*, "Intelligent internet searching agent based on hybrid simulated annealing," *Decision Support Systems,* vol. 28, pp. 269-277, 2000.
[62] Q. Chen, *et al.*, "The design and implement of Internet intelligence agent in electronic commerce environment," in *Computer Supported Cooperative Work in Design (CSCWD), 2010 14th International Conference on*, 2010, pp. 300-305.
[63] A. Abraham and B. Nath, "Hybrid intelligent systems design: A review of a decade of research," *IEEE Transactions on Systems, Man & Cybernetics (Part-C) August,* 2000.
[64] S. Peddabachigari, *et al.*, "Modeling intrusion detection system using hybrid intelligent systems," *Journal of network and computer applications,* vol. 30, pp. 114-132, 2007.
[65] G. Kumar, *et al.*, "The use of artificial intelligence based techniques for intrusion detection: a review," *Artificial Intelligence Review,* vol. 34, pp. 369-387, 2010.
[66] S. Gulpanich, *et al.*, "Distributed control of network devices with remote terminal units," in *IEEE International Conference on Industrial Technology*, pp. 823-828, 2005.
[67] W. Schuler, *et al.*, "A novel hybrid training method for hopfield neural networks applied to routing in communications networks," *International Journal of Hybrid Intelligent Systems,* vol. 6, pp. 27-39, 2009.
[68] R. W. Thomas, *et al.*, "Cognitive networks," in *New Frontiers in Dynamic Spectrum Access Networks, 2005. DySPAN 2005. 2005 First IEEE International Symposium on*, 2005, pp. 352-360.
[69] N. B. Idris and B. Shanmugam, "Artificial intelligence techniques applied to intrusion detection," in *Annual IEEE INDICON*, 2005, pp. 52-55.
[70] R. Copeland, "Network Intelligence-facilitate operators win in mobile broadband era," in *13th International Conference on Intelligence in Next Generation Networks*, pp. 1-6, 2009.
[71] E. Sirin, *et al.*, "Pellet: A practical owl-dl reasoner," *Web Semantics: Science, Services and Agents on the World Wide Web,* vol. 5, pp. 51-53, 2007.
[72] F. Baader and U. Sattler, "An overview of tableau algorithms for description logics," *Studia Logica,* vol. 69, pp. 5-40, 2001.
[73] N. Jaisankar, *et al.*, "Intelligent intrusion detection system using fuzzy rough set based C4. 5 algorithm," in *Proceedings of the International Conference on Advances in Computing, Communications and Informatics*, pp. 596-601, 2012.
[74] J. R. Quinlan, *C4. 5: programs for machine learning* vol. 1: Morgan kaufmann, 1993.

[75] S. S. Khan, *et al.*, "Towards the detection of unusual temporal events during activities using HMMs," in *Proceedings of the 2012 ACM Conference on Ubiquitous Computing*, 2012, pp. 1075-1084.
[76] D. Novikov, *et al.*, "Artificial intelligence approaches for intrusion detection," in *IEEE Long Island Systems, Applications and Technology Conference*, pp. 1-8, 2006.
[77] X. Yao, "Evolving artificial neural networks," *Proceedings of the IEEE,* vol. 87, pp. 1423-1447, 1999.
[78] D. E. Goldberg and J. H. Holland, "Genetic algorithms and machine learning," *Machine learning,* vol. 3, pp. 95-99, 1988.
[79] J. Ferber, *Multi-agent systems: an introduction to distributed artificial intelligence* vol. 33: Addison-Wesley Reading, MA, 1999.
[80] J. Ju, *et al.*, "A Survey on Cloud Storage," *Journal of Computers,* vol. 6, pp. 1764-1771, 2011.
[81] *Apache hbase*. Available: http://hadoop.apache.org/hbase/
[82] H. Hassan, *et al.*, "Towards a framework for evolvable network design," *Collaborative Computing: Networking, Applications and Worksharing,* pp. 390-401, 2009.
[83] M. J. Zaki, "Scalable algorithms for association mining," *Knowledge and Data Engineering, IEEE Transactions on,* vol. 12, pp. 372-390, 2000.
[84] R. Agrawal, *et al.*, "Mining association rules between sets of items in large databases," in *ACM SIGMOD Record*, 1993, pp. 207-216.
[85] J. Case, *et al.*, *A simple network management protocol (SNMP)*: Network Information Center, SRI International, 1989.
[86] T. Finin, *et al.*, "KQML as an agent communication language," in *Proceedings of the third international conference on Information and knowledge management*, 1994, pp. 456-463.
[87] P. D. O'Brien and R. C. Nicol, "FIPA—towards a standard for software agents," *BT Technology Journal,* vol. 16, pp. 51-59, 1998.
[88] D. Steedman, *Abstract syntax notation one (ASN. 1): the tutorial and reference*: Technology appraisals, 1993.
[89] T. Bray, *et al.*, "Extensible markup language (XML)," *World Wide Web Journal,* vol. 2, pp. 27-66, 1997.
[90] A. Farquhar, *et al.*, "The ontolingua server: A tool for collaborative ontology construction," *International journal of human-computer studies,* vol. 46, pp. 707-727, 1997.
[91] Y. C. Sun and G. Clark, "A computational model of an intuitive reasoner for ecosystem control," *Expert Systems With Applications,* vol. 36, pp. 12529-12536, 2009.
[92] Y. Papakonstantinou and V. Vianu, "DTD inference for views of XML data," in *Symposium on Principles of Database Systems: Proceedings of the nineteenth ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems*, 2000, pp. 35-46.
[93] M. Pal and P. M. Mather, "Decision tree based classification of remotely sensed data," in *Paper presented at the 22nd Asian Conference on Remote Sensing*, 2001, p. 9.

[94] D. Rajpathak and R. Chougule, "A generic ontology development framework for data integration and decision support in a distributed environment," *International Journal of Computer Integrated Manufacturing,* vol. 24, pp. 154-170, 2011.
[95] D. L. McGuinness and F. Van Harmelen, "OWL web ontology language overview," *W3C recommendation,* vol. 10, p. 10, 2004.
[96] F. Giunchiglia*, et al.*, *S-Match: an algorithm and an implementation of semantic matching*: Springer, 2004.
[97] J. Ge and Y. Qiu, "Concept similarity matching based on semantic distance," in *Fourth International Conference on Semantics, Knowledge and Grid*, pp. 380-383, 2008.
[98] A. Maedche and V. Zacharias, "Clustering ontology-based metadata in the semantic web," in *Principles of Data Mining and Knowledge Discovery*, ed: Springer, 2002, pp. 348-360.
[99] N. Vasconcelos and A. Lippman, "Statistical models of video structure for content analysis and characterization," *Image Processing, IEEE Transactions on,* vol. 9, pp. 3-19, 2000.
[100] B. Mokhtar*, et al.*, "Network "memory" system for enhanced network services," in *9th International Conference on Innovations in Information Technology (IIT)*, pp. 18-23, 2013
[101] D. Ramage, "Hidden Markov Models Fundamentals," *CS229 Section Notes,* 2007.
[102] K. Seymore*, et al.*, "Learning hidden Markov model structure for information extraction," in *AAAI-99 Workshop on Machine Learning for Information Extraction*, pp. 37-42, 1999.
[103] D. Ourston*, et al.*, "Applications of hidden markov models to detecting multi-stage network attacks," in *System Sciences, 2003. Proceedings of the 36th Annual Hawaii International Conference on*, 2003, p. 10 pp.
[104] J. Yang and Y. Xu, "Hidden markov model for gesture recognition," *Tech. Report CMU-RI-TR-94-10*,1994.
[105] S. Chatterjee and S. Russell, "A temporally abstracted Viterbi algorithm," *arXiv preprint arXiv:1202.3707,* 2012.
[106] W. M. Darling, "A Theoretical and Practical Implementation Tutorial on Topic Modeling and Gibbs Sampling," in *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pp. 642-647, 2011.
[107] G. Casella and E. I. George, "Explaining the Gibbs sampler," *The American Statistician,* vol. 46, pp. 167-174, 1992.
[108] D. Newman*, et al.*, "Distributed inference for latent dirichlet allocation," *Advances in Neural Information Processing Systems,* vol. 20, pp. 17-24, 2007.
[109] D. Sontag and D. M. Roy, "Complexity of inference in latent dirichlet allocation," *Advances in Neural Information Processing Systems (NIPS),* 2011.
[110] K. Henderson and T. Eliassi-Rad, "Applying latent dirichlet allocation to group discovery in large graphs," in *Proceedings of the 2009 ACM symposium on Applied Computing*, pp. 1456-1461, 2009.
[111] B. Mokhtar and M. Eltoweissy, "Biologically-inspired network “memory” for smarter networking," in *8th International Conference on Collaborative*

Computing: Networking, Applications and Worksharing (CollaborateCom), pp. 583-590, 2012
[112] J. Jiang, "Modeling syntactic structures of topics with a nested hmm-lda," in *9th IEEE International Conference on Data Mining*, 2009, pp. 824-829, 2009.
[113] N. Merhav and Y. Ephraim, "Maximum likelihood hidden Markov modeling using a dominant sequence of states," *IEEE Transactions on Signal Processing,* vol. 39, pp. 2111-2115, 1991, doi:10.1109/78.134449.
[114] B. Mokhtar and M. Eltoweissy, "Biologically-inspired Network "Memory" for Smarter Networking," in *International Workshop on Collaborative Big Data (C-Big 2012)*, 2012.
[115] T. T. Nguyen and G. Armitage, "A survey of techniques for internet traffic classification using machine learning," *Communications Surveys & Tutorials, IEEE,* vol. 10, pp. 56-76, 2008.
[116] R. Agrawal and R. Srikant, "Fast algorithms for mining association rules," in *Proc. 20th Int. Conf. Very Large Data Bases, VLDB*, pp. 487-499, 1994.
[117] G. Pengfei*, et al.*, "The enhanced genetic algorithms for the optimization design," in *Biomedical Engineering and Informatics (BMEI), 2010 3rd International Conference on*, 2010, pp. 2990-2994.
[118] K. Lehmann and M. Kaufmann, "6. Random Graphs, Small-Worlds and Scale-Free Networks," *Peer-to-Peer Systems and Applications,* pp. 57-76, 2005.
[119] R. B. Cooper, *Introduction to queueing theory*: Macmillan New York, 1972.
[120] D. Gross*, et al.*, *Fundamentals of queueing theory*: Wiley. com, 2013.
[121] K. Decker*, et al.*, "Middle-agents for the internet," in *INTERNATIONAL JOINT CONFERENCE ON ARTIFICIAL INTELLIGENCE*, 1997, pp. 578-583.
[122] R. Ghaemi*, et al.*, "Evolutionary Query Optimization for Heterogeneous Distributed Database Systems," *World Academy of Science, Engineering and Technology,* vol. 43, p. 2008, 2008.
[123] B. P. Gerkey and M. J. Matarić, "A formal analysis and taxonomy of task allocation in multi-robot systems," *The International Journal of Robotics Research,* vol. 23, pp. 939-954, 2004.
[124] K. W. Tindell*, et al.*, "Allocating hard real-time tasks: an NP-hard problem made easy," *Real-Time Systems,* vol. 4, pp. 145-165, 1992.
[125] M. Kafil and I. Ahmad, "Optimal task assignment in heterogeneous distributed computing systems," *Concurrency, IEEE,* vol. 6, pp. 42-50, 1998.
[126] K. Kogan*, et al.*, "Dynamic Generalized Assignment Problems with Stochastic Demands and Multiple Agent--Task Relationships," *Journal of Global Optimization,* vol. 31, pp. 17-43, 2005/01/01 2005.
[127] H. Hoogeveen*, et al.*, "Non-approximability results for scheduling problems with minsum criteria," *INFORMS Journal on Computing,* vol. 13, pp. 157-168, 2001.
[128] *Genetic Algorithm with Java*. Available: http://kunuk.wordpress.com/2010/09/27/genetic-algorithm-example-with-java/
[129] J. Matyas, "Random optimization," *Automation and Remote Control,* vol. 26, pp. 246-253, 1965.
[130] *Snort*. Available: http://www.snort.org. 2000
[131] *J-Sim, Java-based network simulator*. Available: http://sites.google.com/site/jsimofficial/start-with-j-sim

[132] A. Sobeih, *et al.*, "J-Sim: An integrated environment for simulation and model checking of network protocols," in *Parallel and Distributed Processing Symposium, 2007. IPDPS 2007. IEEE International*, 2007, pp. 1-6.
[133] S. Hettich and S. D. Bay, "The UCI KDD Archive," ed. Irvine, CA: University of California, Department of Information and Computer Science, 1999.
[134] *Winpcap*. Available: http://www.winpcap.org/
[135] B. Caswell and J. Beale, *Snort 2.1 intrusion detection*: Syngress, 2004.
[136] *Autonomous System Taxonomy Repository by CAIDA*. Available: http://www.caida.org/data/active/as_taxonomy/
[137] B. Mokhtar and M. Eltoweissy, "Memory-enabled autonomic resilient networking," in *Collaborative Computing: Networking, Applications and Worksharing (CollaborateCom), 2011 7th International Conference on*, 2011, pp. 132-141.
[138] B. Mokhtar and M. Eltoweissy, "Biologically-inspired network "memory" for smarter networking," in *Collaborative Computing: Networking, Applications and Worksharing (CollaborateCom), 2012 8th International Conference on*, 2012, pp. 583-590.
[139] *An implementation of hidden Markov models in Java (jahmm)*. Available: http://code.google.com/p/jahmm/
[140] G. Arbylon. *Latent Dirichlet Allocation*. Available: http://www.arbylon.net/projects/
[141] F. A. Onat and I. Stojmenovic, "Generating random graphs for wireless actuator networks," in *World of Wireless, Mobile and Multimedia Networks, 2007. WoWMoM 2007. IEEE International Symposium on a*, 2007, pp. 1-12.