Trust Management Systems: Reference Architecture and Personalization

Hisham S. Rashad

Dissertation Submitted to the Faculty of
Virginia Polytechnic Institute and State University in
Partial Fulfillment of
the Requirements for the Degree of

Doctor of Philosophy
in
Computer Engineering

Eltoweissy, Mohamed Youssef (Chair)

Abbott, Amos L
Abdel-Hamid, Ayman Adel
Lu, Chang Tien
Midkiff, Scott F
Xuan, Jianhua

July 27, 2017

Blacksburg, Virginia

Keywords: Trust Management, Reputation Management, Trust Personalization, Trust Management Reference Architecture, Trust-driven Services, Human Inspired Trust Management.

Trust Management Systems: Reference Architecture and Personalization

Hisham S. Rashad

Abstract

Trust is the cornerstone of success in any relationship between two or more parties. Generally, we do not socialize, seek advice, consult, cooperate, buy or sell goods and services from/to others unless we establish some level of mutual trust between interacting parties. When ecommerce was merging infancy, the concept of trusting an entity in a virtual world was a huge obstacle. Gradually, increasingly-sophisticated, largely generic reputation scoring and management systems were embedded into the evolving marketplaces. Current technologies to include cloud computing, social networking, and mobile applications, coupled with the explosion in storage capacity and processing power, are evolving large-scale global marketplaces for a wide variety of resources and services, such as Amazon.com, BitTorrent, WebEx and Skype. In such marketplaces, user entities, or users for short; namely, consumers, providers and brokers, are largely autonomous with vastly diverse requirements, capabilities and trust profiles. Users' requirements may include service quality levels, cost, ease of use, etc. Users' capabilities may include assets owned or outsourced. Trustors' profiles may include what is promised and commitments to keep these promises. In such a large-scale heterogeneous marketplace, the trustworthy interactions and transactions in services and resources constitute a challenging endeavor

Currently, solving such issues generally adopts a "one-size fits all" trust models and systems. Such approach is limiting due to variations in technology, conflicts between users' requirements and/or conflicts between user requirements and service outcomes. Additionally, this approach may result in service providers being overwhelmed by adding new resources to satisfy all possible requirements, while having no information or guarantees about the level of trust they gain in the network. Accordingly, we hypothesize the need for personalizable customizable Trust Management Systems (TMSs) for the

robustness and wide-scale adoption of large-scale marketplaces for resources and services. Most contemporary TMSs suffer from the following drawbacks:

- Oblivious to diversities in trustors' requirements,
- Primarily utilize feedback and direct or indirect experience as the only form of credentials and trust computations,
- Trust computation methodologies are generally hardcoded and not reconfigurable,
- Trust management operations, which we identify as monitoring, data management, analysis, expectation management, and decision making, are tightly coupled. Such coupling impedes customizability and personalization, and
- Do not consider context in trust computations, where trust perspectives may vary from a context to another.

Given these drawbacks and the large scale of the global marketplace of resources and services, a reference architecture for trust management systems is needed, which can incorporate current systems and may be used in guidance and development of a wide spectrum of trust management systems ranging from un-personalized to fully personalized systems. Up to our knowledge, no TMS reference architecture exists in the literature.

In this dissertation, we propose a new Reference Architecture for Trust Management (RATM). The proposed reference architecture applies separation of concern among trust management operations; namely, decision expectation, analytics, data management and monitoring. RATM defines trust management operations through five reconfigurable components which collectively can be used to implement a wide spectrum of trust management systems ranging from generic to highly personalized systems. We used RATM for trust personalization, where we propose a Personalized Trust Management System (PTMS) based on RATM. We evaluated PTMS's scalability and demonstrated its effectiveness, efficiency and resilience by contrasting against a Generic Trust Management System (GTMS). We used two case studies for our evaluations; namely, BitTorrent and a video conferencing application.

Intellectual Merit

In this work, we propose RATM, a reference architecture for trust management systems that can be used to implement a wide variety of trust management systems ranging from generic systems (un-personalized) to highly personalized systems. We assume service-based environment where consumers, brokers and service providers are interacting and transacting in services and resources to satisfying their own trust

requirements. We used RATM to implement a personalized trust management system (TMS). The main contributions of this work are as follows:

- Proposing RATM for the guidance and development of a wide spectrum of TMSs ranging from un-personalized to fully personalized systems, and
- Utilizing our RATM to propose and implement a personalized, scalable TMS with varying trust computation models.

Broader Impact

RATM provides reference architecture for trust management which can be used to develop and evaluate a wide spectrum of TMSs. Personalization, in general, paves the road for reaching high levels of satisfaction, where interacting parties' requirements are individually considered and thus consumers are served the best suited service(s). Hence, we claim that PTMS would largely enhance large-scale heterogeneous systems offering services and resources. This could lead to more cooperation, more transactions, more satisfaction, less malicious behavior and lower costs.

Trust Management Systems: Reference Architecture and Personalization

Hisham S. Rashad

General Audience Abstract

Trust is the cornerstone of success in any relationship between two or more persons. Generally, we do not socialize, seek advice, consult, cooperate, buy or sell goods and services from/to others unless we establish some level of mutual trust between interacting parties. When ecommerce was firstly used, the concept of trusting a service delivered by someone who is not physically in the same place was a huge obstacle. Gradually, more sophisticated, largely generic reputation scoring and management systems were used into the new internet marketplaces. A reputation scoring and management system is a system which collects feedback from different users about service providers in a certain marketplace on the internet and uses them to anticipate future behavior of these providers. Current computer technologies to include cloud computing, social networking, and mobile applications, coupled with the explosion in computer and mobile devises' storage capacity and processing power, are evolving large-scale global marketplaces offering a wide variety of resources and services to consumers across the globe. Examples include Amazon.com, BitTorrent, WebEx and Skype. In such marketplaces, consumers, providers and brokers, are largely autonomous with vastly diverse requirements, capabilities and trust profiles. By autonomous we mean acting in accordance with one's moral duty rather than one's desires. Users' requirements may include service quality levels, cost, ease of use, etc. Users' capabilities may include assets owned or leased from others. Trustors' profiles may include what is promised and commitments to keep these promises. In such a large-scale marketplace, the trustworthy interactions and transactions in services and resources constitute a challenging endeavor. By trustworthy interaction we mean transactions which deliver results that are accepted by all parties.

Currently, solving such issues of trust generally adopts a "one-size fits all" trust models and systems. By trust models and systems we mean computer programs which perform the reputation scoring and management. i.e. select a single service which can serve all requirements. Such approach is limiting due

to variations in technology, conflicts between users' requirements and/or conflicts between user requirements and service outcomes. Additionally, this approach may result in service providers being overwhelmed by adding new resources to satisfy all possible requirements, while having no information or guarantees about the level of trust they gain in the eye of their consumers.

Accordingly, we hypothesize the need for personalizable customizable Trust Management Systems (TMSs) for the robustness and wide-scale adoption of large-scale marketplaces for resources and services. In other words, we assume the need for a trust management system which can select services satisfying transaction parties' requirements. Most contemporary TMSs suffer from the following drawbacks:

- Select one size fits all service,
- Utilize one and only one type of data for calculating the score used for anticipating the future behavior of a party,
- Utilize one and only one method to calculate the score value used for anticipating the future behavior of a party,
- Trust scoring calculation method does cannot be reprogrammed,
- Trust scoring calculation method does not consider the context in which the data was collected.

Given these drawbacks and the large scale of the global marketplace of resources and services, a reference architecture for trust management systems is needed, which can incorporate current systems and may be used in guidance and development of a wide spectrum of trust management systems ranging from un-personalized to fully personalized systems. Up to our knowledge, no TMS reference architecture exists in the literature.

In this dissertation, we propose a new Reference Architecture for Trust Management (RATM), which overcomes the drawbacks of current systems. It proposes evaluating trust by number of flexible operations namely, decision expectation, analytics, data management and monitoring. These operations collectively can be used to implement a wide spectrum of trust management systems ranging from generic to highly personalized systems. We used RATM for trust personalization, where we propose a Personalized Trust Management System (PTMS) based on RATM. We evaluated PTMS's ability to sustain the increasing number of users and demonstrated its effectiveness, efficiency and its ability to resist attacks. We achieved that by contrasting experimentation results against that of a Generic Trust Management System (GTMS). We used two case studies for our evaluations; namely, BitTorrent and a video conferencing application.

Intellectual Merit

In this work, we propose RATM, a reference architecture for trust management systems that can be used as a guide in the implementation of a wide variety of trust management systems ranging from generic systems (un-personalized) to highly personalized systems. We assume service-based environment where consumers, brokers and service providers are interacting and transacting in services and resources to satisfying their own trust requirements. We used RATM to implement a personalized trust management system (TMS). The main contributions of this work are as follows:

- Proposing RATM for the guidance and development of a wide spectrum of TMSs ranging from un-personalized to fully personalized systems, and
- Utilizing our RATM to propose and implement a personalized, scalable TMS with varying trust computation models.

Broader Impact

RATM provides reference architecture for trust management which can be used to develop and evaluate a wide spectrum of TMSs. Personalization, in general, paves the road for reaching high levels of satisfaction, where interacting parties' requirements are individually considered and thus consumers are served the best service which satisfy his/her requirements. Hence, we claim that PTMS would largely enhance large-scale heterogeneous systems offering services and resources. This could lead to more cooperation, more transactions, more satisfaction, less malicious behavior and lower costs in such market places.

Dedication

To all my family especially, mom, dad, sister, brother and my amazing wife for all their continuous prayers, encouragement, support and love.

To my lovely children: Tarek and Omar, the sweet adorable light of my life.

Acknowledgements

Above all, all praise and thanks are due to Allah (God), the almighty for the uncountable blessings that He has bestowed upon me, and for granting me the strength, the patience and the capability to complete this work successfully. I also thank Allah for blessing me with many great people who without their contribution, support, guidance, encouragement and prayers, this work would not have been. I cannot hope to repay them in kind, and ask Allah to repay them all.

First, I would like to express my gratitude and gratefulness to my advisor Prof. Mohamed Eltoweissy. Working with him has been a great learning experience. His permanent guidance, support and encouragement have been constant in this journey, and were essential to bring this work out to light. He has been the source of countless good research ideas, and at the same time his feedback has greatly improved my research, papers, and talks. Without his mentoring, I wouldn't be where I am today.

I also would like to thank the members of my committee, Prof. Scott Midkiff, Prof. Jason Xuan, Prof. Chang-Tien Lu, Prof. Lynn Abbott, and Prof. Ayman A. Abdel-Hamid for their generosity in taking the time to review and offer insightful suggestions to greatly improve this dissertation.

Secondly, I would like to provide special thanks to prof. Sedki Riad and prof. Yasser Hanafy for their dedicated work in establishing VTMENA program which provided me the opportunity to have my PhD from a prestigious university as Virginia Tech.

Thirdly and most importantly, I would like to thank all my family members; my parents, my brother, my sister and my auntie for their continuous encouragement and prayers. My parents in law, for the unfailing confidence and encouragement they have given me. Also, Tarek and Omar, my sons, whose smiles were the motive behind overcoming all the challenges and obstacles I faced. Finally, my wife, who provided me with the quiet and peaceful environment in which I achieved my work and for her support, encouragement, continuous prayers, and for all the smiles she put on my face during the stressful moments in my journey.

Table of Contents

List of Figures

List of Tables

Chapter 1

1. Introduction

1.1. Motivation and Problem Statement

Trust is the cornerstone of success in any relationship between two or more parties. It is the key enabler of interactions, where we cannot socialize, seek advice, consult, cooperate, buy or sell goods and services from/to others unless we establish some level of mutual trust between interacting parties.

When ecommerce was emerging, the concept of trusting an entity in a virtual world was a huge obstacle. At that time, service consumers did not have tangible aspects upon which trust can be built. While service providers had no means of identifying who they are delivering to, and how it would affect their reputation in the marketplace. Gradually, increasingly-sophisticated, largely generic reputation scoring and management has been embedded into the evolving marketplaces. However, the emergence of cloud computing, social networking, and mobile applications coupled with the explosion in storage and processing power, enhanced services ubiquity, availability and mobility, while drastically reduced the cost of computing and communications. Such enhancements gradually formed a global marketplace for a wide variety of resources and services including, merchandise marketplaces, such as Amazon.com, ebay.com and Walmart.com, file sharing marketplaces such as BitTorrent [42] and Gnutella [43] network and Cloud video conferencing services such as Adobe Connect, WebEx and Skype. In such a large-scale marketplaces, user entities, or users for short, which include: roles of consumers, roles of providers and roles of brokers, are largely autonomous with vastly diverse requirements, capabilities and trust profiles. Users' requirements may include service quality levels and fees needed. Users' capabilities may include assets owned or outsourced. Trustors' profiles may include what is promised and commitments to keep these promises.

Given such large-scale heterogeneous marketplace, the trustworthy interactions and transactions in services and resources constitute a challenging endeavor. Here, we define trust of a party in another party as: the belief or disbelief that another party, for a said subject of trust in a given context, has the ability to exhibit a set of acceptable actions in the future, for the welfare of the trusting party.

Given this perspective, we hypothesize the need for a personalizable customizable trust management system for the robustness and wide-scale adoption of such large-scale marketplaces for resources and services.

In the absence of personalization of trust, users' trust and service providers utilize "one-size fits all" or average service to satisfy their requirements. However, in some cases, such service may not exist and even would be hard to realize. In addition, users' requirements may contradict with the one-size service outcomes, resulting in an unrealistic situation, for instance, an internet user may require only a reliable network service with low cost. In this case, un-personalized trust management systems would recommend the highest trust-score/reputable service, delivering overall high QoS parameters, having the assumption that it would be the most satisfying. Other service alternatives would have less chance of being recommended despite the fact that they deliver reliable service at lower costs.

Additionally, service providers may be overwhelmed by adding new resources to satisfy all possible requirements, while having no information or guarantees about the level of trust they gain.

In literature, a plethora of trust management systems with varying degrees of success have been proposed; however, many suffer from several drawbacks including:

1. Most existing trust management systems are oblivious to the diversity in user trust. Regardless of each user trust's including requirements, trust profile or capabilities, a single evaluation score formed by feedbacks from heterogeneous resources is used to identify suitable services.
2. Trust management systems are oblivious to mutual trust requirements, where most existing works focus on trust issues with service providers only, while neglecting the need to also trust the consumer.
3. Most systems do not ensure the credibility of trust data, where few systems factor the collected feedbacks by reliability of feedback provider, or degree of trustworthiness in its source.
4. Most systems do not consider context in trust computations, where trust may vary from one context to another. For instance, one may trust a reputable organization as Apple for their computers, tablets and mobile phones, while we may not provide the same level of trust when considering an autonomous vehicle, given other companies as Nissan, Mercedes and Audi.
5. Most systems primarily utilize feedback and direct or indirect experience as the only form of credentials and trust computations, while other forms are hardly ever utilized. Other forms may include for instance: popularity of service provider, warranty, guarantees, and certifications (more data forms to be found in chapter 2).

6. Trust as well as trust-based decision support computations and methodologies are generally hardwired and not reconfigurable. In such case, derivation of trust cannot reflect requestors' point of view. For instance, a file storage service may have users requiring frequent access while others requiring high performance. In the first case, a trust score which reflects availability of the service is needed, while in the second case a trust score which reflects response time and bandwidth is needed.
7. Trust management operations are tightly coupled. Such coupling hinders reusability of existing components from current applications, such as data management, analysis and monitoring. In addition, it prevents utilization of any reconfigured components resulting in less generic and personalizable operations. Moreover, it exposes trust data to all trust management operations, resulting in potential violation of trust data privacy and vulnerabilities that may violate trust data confidentially and integrity.

Given these drawbacks and the large scale of the global market place of resources and services, a reference architecture for trust management is required, for the guidance and development of a wide spectrum of trust management systems ranging from un-personalized to fully personalized systems. Up to our knowledge, no reference architecture exists in the literature for trust management.

1.2. Scenario

To motivate our work through the rest of the document, we will refer to the following two scenarios:

1.2.1. BitTorrent File Sharing

Consider a BitTorrent [42] file sharing swam in which leechers are contributing to gaining high bandwidth in order to speed up their downloads. By Leechers, we mean peers downloading and uploading chucks they received so far from the swarm. In this network, selfish misbehaving leechers which are trying to poison the download of others. Typically, they send fake chunks of the shared file to other leechers in order to increase their bandwidth they get from them. These selfish misbehaving leechers have the goal of gaining unfairly high bandwidth in order to speed up their download. Here, we should note that selfish misbehaving leechers do not continuously send bogus chunks; rather, they may send valid chucks if they have it to other leechers. This means that they somehow can be useful.

On the other hand, well behaving leechers vary among their tolerance towards receiving bogus chunks. Some are highly tolerant while others are not. For instance, some may accept 2 to 5 bogus chunks, while others may accept up till 10.

Figure 1.1 illustrates the scenario. In the figure, we have a common seeding node and number of leechers with different colors. The seeding node represents the node sharing the complete file. Nodes of different outline colors represent varying degrees of tolerance towards bogus chunks.

The red leecher represents a misbehaving node that is exploiting its download links from other leechers and sending bogus chunks.

Given this scope, it is required to help leechers optimize their selections among other leechers and gain maximum benefit of misbehaving leechers, without violating leechers tolerance towards bogus chunks. A trust management system is needed to execute this role. Current trust management systems fail to provide a comprehensive solution which adopts the diverse requirements of leechers. Additionally, "the one size fits all" solution provided by current systems may not maximize the benefit from misbehaving leechers.

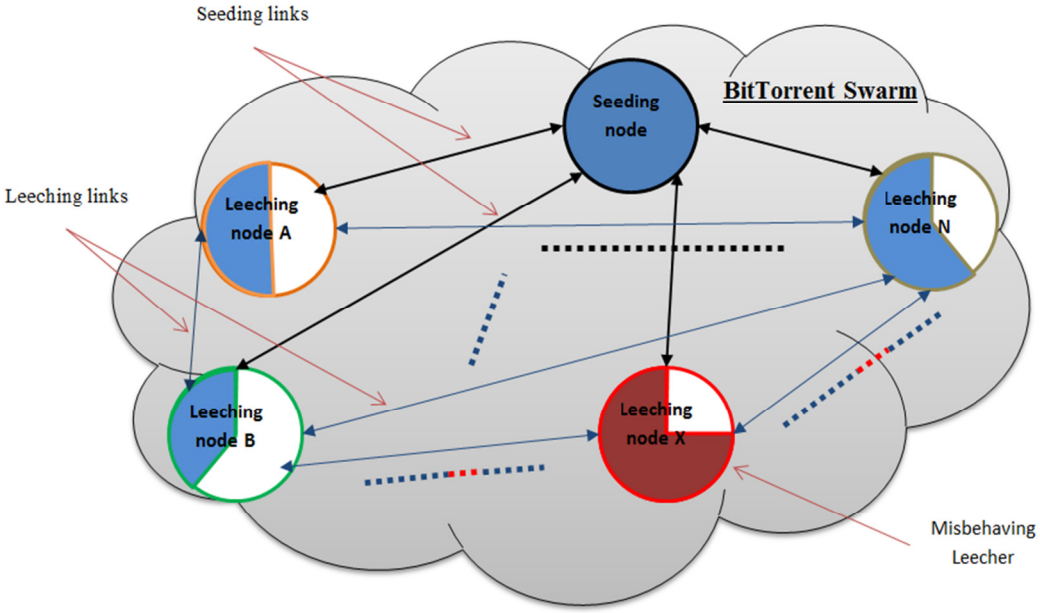


Figure 1.1 BitTorrent file sharing scenario.

1.2.2. Video Conferencing

A number of research institutes having groups of researchers (professors and students), who are conducting a number of video conference meetings to discuss and share their work with their colleagues and/or professors. A number of participants in a conference meeting may vary, starting from 2 participants and increasing as required. Participants location may also vary where some may conduct their

meetings from inside their institute while others may attend from outside. The devise used by which each participant, who attends his conference meeting, may also vary. Some may attend using PCs while other may use their smart phones or tablets.

Given this scope, video senders and receivers (participants) are diverse among their requirements including: video streaming rates, video resolution and video compression.

Streaming rates

In some cases, video stream is achieved at the busiest time slots of the day, hence management of video stream sending rates is required to manage for congestions in the network. Other video stream senders may operate in less congested hours of the day.

Resolution

Some video conferences may include sharing of figures and graphical content, hence requiring high resolution video quality, while others may include only discussions, hence, low video resolution may be acceptable.

Compression

During busy time slots of the day, reducing packet sizes may be a useful approach to deal with congested networks. In addition, some video stream senders may stream videos from their mobile phones or tablets which operate on limited battery resources. Such limited resources place restrictions on type of video compression that can be utilized by the video stream service. Other senders may operate from their PCs and thus may not have any limitations.

Each participant in the conference/meeting is a video sender and a receiver at the same time (figure 1.2) forming a peer-to-peer overlay network of senders and receivers. A video sender utilizes a video stream service to create and send streaming video he/she requires to their conference participant(s).

The streaming service outcome is evaluated using three parameters; namely, video resolution, video compression and video streaming rate. The video resolution is the number of pixels per video frame generated by the streaming service. The video compression is the amount of frame size reduced by the streaming service. While video streaming rate is the sending rate of video stream packets. The sending rate here depends on congestion observed in the network, where the service adjusts its sending rate

(increase or decrease) according to the number of packets lost in the network (more details can be found in section 5.2). Figure 1.2 illustrates the scenario. In the figure, each conference participant has a different color representing different requirements than his/her counterpart. Each participant may be in a specific location, using specific device and at a certain point in the day resulting in different requirements for each participant.

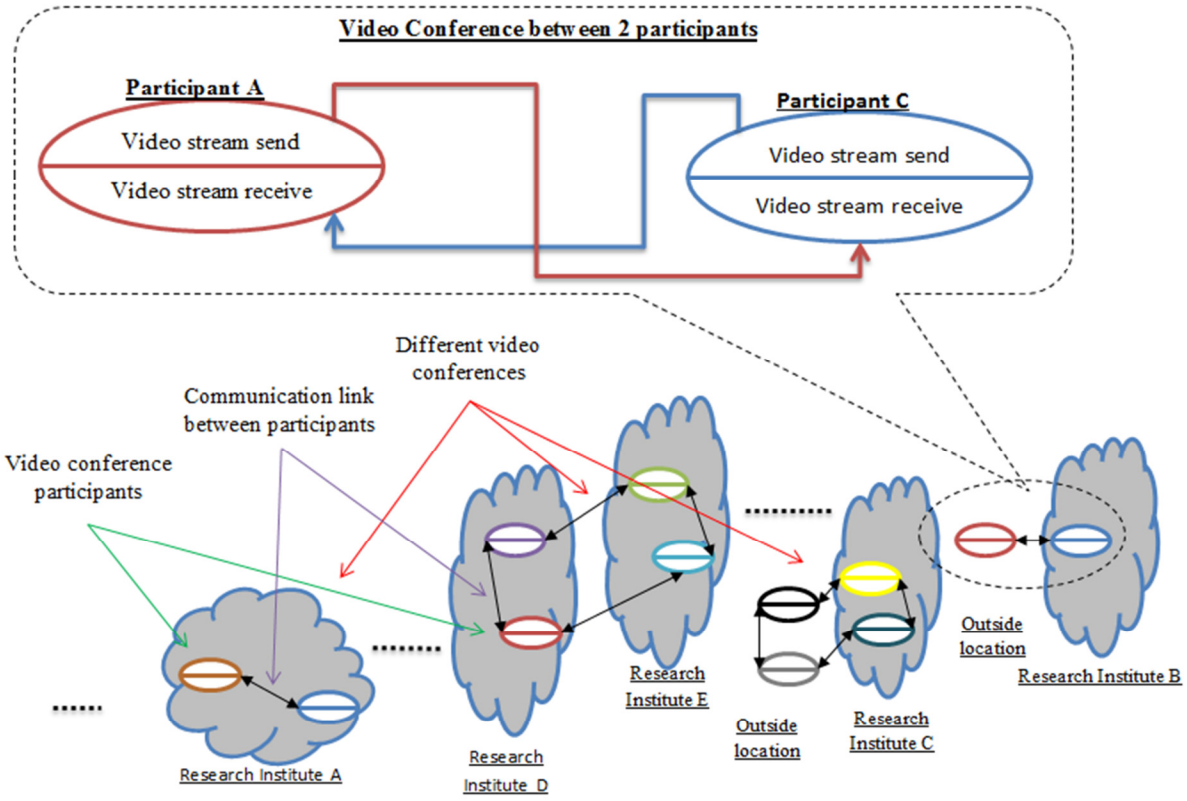


Figure 1.2 Video conferencing scenario.

In our scenario, we have a number of video stream service alternatives that are diverse among their outcomes such that each requirement may find at least one service alternative which may satisfy it. These services are available to all participants to choose among them without any restrictions. Accordingly, for each conference participant, it is required to choose the suitable streaming service alternative which satisfies his/her requirements.

If we consider current systems, we may find that they adopt a “one size fits all” solution, where they recommend service providers whose average performance and behavior are accepted by the majority of

users. Here, majority acceptance does not necessarily reflect total satisfaction; on the contrary, it may indicate that only the major part of requirements is met while, still some other requirements are not satisfied. On the other hand, the best fit choice would maximize conference participants' likelihood of success and satisfaction.

1.2.3. Goals

Our main goal in this research is to achieve effective and efficient trust based transactions in services and resources in a large scale heterogeneous marketplace. To achieve this goal, we develop a reference architecture to be used in guidance and development of a wide spectrum of trust management systems ranging from un-personalized to fully personalized systems. Then, we develop a personalizable customizable scalable trust management system using our proposed reference architecture for the robustness and wide-scale adoption of large-scale marketplaces for resources and services. We address scalability and attack resilience in our proposed system as they pose significant challenges for large-scale personalizable trust management.

1.3. Research Approach

In this work, we explore and enable trust personalization and provide trust management solutions in the form of trust management services. To accomplish this, we started by a series of classifications for trust/reputation management systems depicting different trust computations, different forms of trust data, different trust management architectures and finally programmability of trust management. Based on these classifications, we started identifying common features, less common features, gaps in research and limitations of current systems.

Based on limitations of the current systems, we develop a Reference Architecture for Trust Management (RATM). The proposed reference architecture applies separation of concern among trust management operations: decision, expectation, analysis, data management and monitoring. In addition, it defines trust management operations through five reconfigurable components which collectively can be used to implement a wide spectrum of trust management systems ranging from generic to personalized trust management systems. We used RATM for trust personalization, where we propose a Personalized Trust Management System (PTMS) based on RATM. In our evaluations, we computed time and space complexities of PTMS's. In addition, via simulation, we evaluated PTMS's scalability, effectiveness, efficiency and resilience. We evaluated scalability of PTMS by contrasting its' results metrics when

increasing nodes population. We evaluated effectiveness and efficiency of PTMS by comparing its results metrics against that of a Generic Trust Management System (GTMS). The GTMS is a feedback scoring system for trust evaluation of services. We focused in our work on large-scale heterogeneous marketplaces, where different roles (consumers, brokers and providers) are using trust-driven transactions on services and resources that would satisfy their requirements. Within our work, we addressed several questions including:

- What are the foundational trust management operations necessary to build any trust management system?
- What are the foundational components of trust management suitable for large-scale service-based heterogeneous marketplaces?
- How to achieve personalization of trust?
- How to scale a personalized trust management system?
- Will personalization enhance trust management in large-scale heterogeneous marketplaces of services and resources?

RATM comprises a set of reconfigurable components that can be used to virtually implement any trust management system. Trust requirements of each user are expressed using Trust Management Definition Language (TDL). RATM components represent operations of trust management, which includes decision, expectation, analytics, data management and monitoring. Separation of concern among these operations provides the flexibility of reuse from existing applications, where analysis, monitoring and data management services from existing applications can be incorporated and used in our system. Additionally, the separation of concern enables the utilization of customizable, personalizable operations of the system.

Hypothesis

Personalization of trust management would enhance users' satisfaction and transactions in large-scale marketplaces of services and resources, given the vastly diverse requirements, capabilities and trust profiles.

The following constitutes an effective and efficient way for personalization of trust management:

- Separation of concerns among trust management operations.

- Enabling users to express and manage their trust requirements.
- Enabling reconfiguration and coexistence of diverse trust computations.
- Scalable highly distributed/federated architecture.
- Sampling of trust data using clustering techniques to reduce complexity and support scalability.

1.4. Evaluation

Our approach, evaluation is based on contrasting effectiveness and efficiency of PTMS and GTMS implemented in a heterogeneous simulated environment of resources and services. We used ns2 [30] network simulation environment in our work, where we simulated the two scenarios illustrated in sections 1.2.1 and 1.2.2.

We used two case studies for our evaluations, including: BitTorrent and Video conferencing application. In **BitTorrent case study**, we contrasted simple version of PTMS which adapts according to preferences of peers in the network against a number of simple GTMSs, each of fixed thresholds matching peers' preferences in the network. Evaluations provided by PTMS or GTMS help leechers to select the suitable leeching service and optimized use of selfish misbehaving leechers in network. We experimented for only high diversities in trust preferences. Diversities in trust preferences are the ratio between the number of trustors' unique preferences and the number of service alternatives serving them (more details in section 5.2.3.1.1). Our experiments' results showed that:

- The utilization of PTMS enhanced leechers' QoS level of satisfaction: PTMS produced the results which, when compared to that of GTMS versions, constituted a good compromise in terms of overhead, exposure duration, wasted and success ration, for all leechers given their diverse trust thresholds.
- Trust personalization increases resiliency to attacks: with increased amount of malicious behavior in the network, effectiveness and efficiency results for PTMS outperformed that of GTMS for various degrees of maliciousness.

In **Video conferencing case study,** we contrasted a more sophisticated version of PTMS against a simple feedback scoring version of GTMS. The GTMS selects an average service alternative in network based on

trustors' feedbacks. We contrasted effectiveness and efficiency metrics for both systems and used ns2 [30], where we modified the default UDP agent structure to implement the five components of RATM (more details to be found in section 5.2). Additionally, we evaluated PTMS's scalability, where we contrasted number of messages generated by the system and the amount of stored data, for different nodes population.

Unlike BitTorrent case study, we experimented for three different diversities in trust preferences, including low, medium and high. Our results showed that:

- For high and medium diversities, PTMS produced better effectiveness, efficiency and resilience results.
- While for low diversities, PTMS produced almost the same results as the un-personalized system.
- For low, medium and high diversities, PTMS showed scalable results, including decrease in stored data and linear increase in total number of messages.

We proposed using two trust computation models; namely, human inspired trust computation and statistical trust computation. The first computation model includes four parameters inspired by a human trust model; namely, Intent, Integrity and Results, while the second computation model is based on statistics of simple network parameters. For evaluating flexibility of PTMS, we utilized both trust computations in a single PTMS and compared effectiveness and efficiency results for both computations. The results showed:

- Close effectiveness, efficiency and resilience values among the two trust computations.
- The utilization of the human trust model trust computation parameters, including Intent and Integrity, enhances efficiency and resilience metrics values given a relatively small history window size.

1.5. Contributions

Intellectual Merit

In this work, we propose RATM, reference architecture for trust management. RATM comprises a number of reconfigurable components that can be used to implement a wide variety of trust management system, ranging from traditional systems (un-personalized) to personalized systems. We assume service-based environment where consumers, brokers and service providers are interacting and transacting in

services and resources to satisfy their own trust requirements. We used the reference architecture to implement a personalized trust management system. The main contributions of this work are as follows:

- Proposing RATM for the guidance and development of a wide spectrum of trust management systems ranging from un-personalized to fully personalized systems.
    - RATM decouples different trust management operations: decision, expectation, analytics, data management and monitoring.
    - RATM comprises a number of distributed reconfigurable components which can be configured to coordinate and communicate with its counterparts to perform trust management operations.
    - In addition to effectiveness and efficiency, the proposed reference architecture targets resiliency against attacks, extensibility, and scalability.
    - RATM may provide insights for the study of trust and reputation management.
- Proposing fully distributed personalized scalable trust management system based on RATM.
- Defining and quantifying trust in terms of four "quantifiable" parameter sets denoting: Intent, Integrity, Capability and Results.

We evaluated trust personalization using simulation, where we used RATM to implement personalized and generic trust management systems.

Our research findings are as follows:

1. Personalization produces better effectiveness, efficiency and resilience values than current approaches (un-personalized), given high and medium diversities of trust preferences.
2. Given low diversities of trust preferences, personalization produces same results as the current approaches (un-personalized).
3. Personalization endures high time and space complexities when compared to current (un-personalized) approaches, hence a tradeoff is required when choosing among personalization and current approaches (un-personalized).
4. Trust data sampling using clustering techniques enhances scalability of the personalized system.
5. The utilization of the proposed human-inspired trust model trust computation enhanced effectiveness and resilience of the system.

Broader Impact

Trust is the key enabler of interactions, where we cannot socialize, seek advice, consult, cooperate, buy or sell goods and services from/to others unless we establish mutual trust with our interaction parties and our interactions. RATM provides reference architecture for trust management which can be used to implement a wide spectrum of trust management systems ranging from personalized to un-personalized systems. Personalization, in general, paves the way for reaching high levels of satisfaction, where interaction party's requirements are individually considered and thus consumers are served the best suited service(s). Hence, we claim that PTMS would largely enhance large-scale heterogeneous systems offering services and resources. This leads to more cooperation, more transactions, more satisfaction, less malicious behavior and lower costs would occur.

In addition, RATM may be used for the design and implementation of trust management systems in large scale network environment such as PlanetLab [41]. In this scope, the deployed trust management system and its application would serve as a real world test bed for studying trust and reputation management. In such environment, scenarios of real world may be studied and new forms of trust computations can be tested and evaluated.

1.6. Document Organization

The remainder of this dissertation is organized as follows. Chapter 2 presents related work identifying strengths and weaknesses of each work through a number of general classifications of reputation/trust management systems. This includes different forms of trust computations, different forms of trust data, different forms of trust management systems architectures and a classification of different forms of programmability of trust. Through these classifications, we provide an overview and analysis identifying the gaps in literature. Following that, we provide an overview of BitTorrent file sharing protocol. In addition to, a brief survey for some clustering techniques, including Affinity Propagation, K-means, K-medoids and the density based clustering technique DBSCAN. Chapter 3 includes the proposed reference architecture and its working environment, along with its components' function, structure, sub-structure, operation and personalization factors. Chapter 4 includes the proposed personalized trust management system, illustrating its structure, operation, trust computations used and scalability of the system. Chapter 5 presents our evaluations, including experiments, results and their analysis. In addition, a brief

discussion on, when to choose a personalized system rather than a tradition system (un-personalized). Finally, chapter 6 provides a conclusion and outlines directions of future work.

Chapter 2

2. Related Work and Background

A plethora of trust management systems have been proposed in contemporary literature. In this chapter, we provide an overview and analyses of related works identifying their strengths and weakness. Our presentation is based on four main aspects depicting four classifications, including trust computation, trust data, trust architecture and trust programmability. By trust computation, we mean the process of producing trust evaluations. This includes the operations applied to data in order to generate an evaluation. By trust data, we mean all forms of data involved in trust computation. By trust architecture, we mean trust management systems architecture. By trust programmability, we mean the degree of customizability of trust management system in terms of producing trust evaluations which may satisfy different trust requirements. These four aspects of trust were based on our study of literature, where we found that different contributions from different works may fall into one of these four categories. Accordingly, for each aspect, we built a classification based on different works in literature, identifying what was and what was not achieved by current systems. In the scope of these classifications, we present our analysis of related works, identifying common features, less common features, limitations and gaps.
In addition to related works, presentation and analysis, we provide an overview of BitTorrent file sharing protocol [42]. Moreover, we provide a brief survey of some clustering techniques, identifying pros and cons and usefulness of each from our work's perspective. This includes: Affinity Propagation [31], K-means [53], K-medoids [54], and DBSCAN [55].

2.1. Related Work

Trust can be viewed from multiple dimensions. We enclose four main dimensions which describe trust; namely computation, data, system architecture and degree of personalization.

2.1.1. Classification of Trust Computation

Trust computations can be classified into three main sub-classes (figure 2.1); namely, Experience-based, Credential-based, and Popularity-based. A hybrid class also exists.

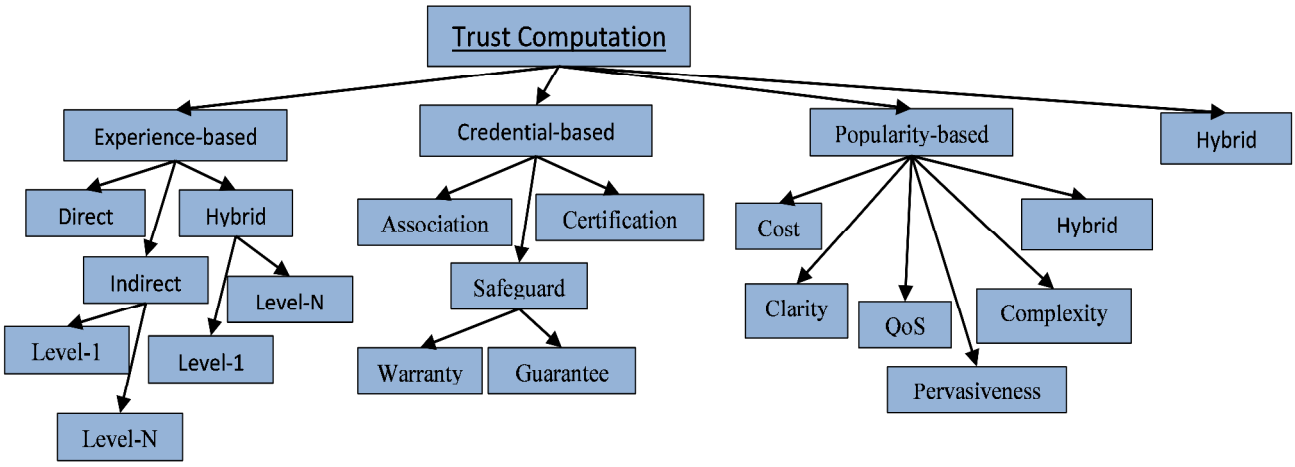


Figure 2.1 Different forms of trust computations.

2.1.1.1. Experience-based

The Experience-based class represents systems computing trust based on local or remote information (opinions) from parties who had one or more experience(s) with the trustee[1]. A trustee may be a service consumer, a service provider or a third party. Opinions may be direct, indirect or hybrid. In other words, information sources who are asked about their opinions, may or may not have experience with the trustee. Direct opinion represents systems computing trust based on self-experience only. It doesn't involve any external parties.

In indirect opinion systems computing, trust is based on recommendation(s), to or not to deal with the trustee (Level 1 Experience), from experienced parties, while experience-less parties will refer to another party who he/she thinks have experience with the trustee (Level N Experience). For example, if we consider having 4 software agents; A, B, C, and D. Agent A requires service X from agent D. Agent B, at some time in the past consumed service X from agent D, while agent C trusts B and knows that it has an

[1] A **trustee** is the person, software agent or an object in which trust is established, while a **trustor** is the person, software agent or an object which establishes trust.

experience with agent D. Given that A trusts C, A contacts C to ask about service X provided by D. Since C does not have any experience with D he/she recommends A to ask B about his direct experience with D. Experience-based hybrid systems utilize Direct in addition to Level 1 Experience and Direct in addition to Level N Experience.

2.1.1.1.1. Direct Opinions

The following related works represent direct opinions.

[7 and 8] proposed trust management systems for distributed file sharing applications. In [7], the proposed system collects 2 types of feedback, the first is concerned about evaluating QoS delivered by peers. The second is concerned about usefulness of provided feedback. Trust is calculated by summation of all evaluation feedbacks weighted by usefulness of each. The work, via simulation, presented a scalability study as well as system's robustness against a variety of attacks by malicious users. The work only isolated non cooperative nodes, while it did not provide incentives for cooperativeness. Additionally the work was oblivious to forms of misbehavior, such as colluding and blackmailing. Moreover and most importantly the P2P model had all data distributed across the network having no single peer viewing the complete global view of reputation and causing considerable amount of traffic for data allocation. In [8], as [7] the system collects 2 types of feedback: the first is concerned about evaluating QoS delivered by trustee; the second is concerned about usefulness of provided feedback. Basically, the work proposes a light weight middle ware solution to prevent tampering with local stored and transmitted data. However, the work adopted a distributed storage which causes considerable amount of traffic for data allocation.

[2, 3 and 23] proposed trust management systems for Distributed systems. The systems solicit feedback from peers after each transaction. However, [2] recognized trust in a peer as average feedback, while [3] factored the averaged feedbacks by trustor's direct experience with the feedback provider. The work presented in [2] proposed using a partially distributed solution to calculating trust. Additionally, it provides a public key based mechanism that periodically updates the peer reputations in a secure, light-weight fashion. However, solution trades some accuracy in the reputation score computations to keep the overheads of the system to a minimum. The work presented in [3] focuses on anonymity as a solution against tampering attacks and proposes a trust based protocol which provides anonymity for both the trust host and the trust querying peer. The work again adopts a distributed storage mechanism and thus affects traffic in the network. Additionally, the work lacked a study showing system's performance given different forms of attacks. In [23] the proposed system utilizes direct experience with other peers in the

network and updates it with time. Direct experience is used to build a multi directed graph representing trust among peers in the network. The graph edges represent positive values of trust, while no edges represent unknown trust or un-trust. The main strength of this work is the improvement of overhead of computation and increase the reliability of the system. The work only focused on these points and did not consider other important issues such as resilience and/or security of the system.

[11 and 12] proposed a reputation management system for BitTorrent networks. In [12], each Leecher after completing a chunk, provides: -1 feedback if the received chunk is bogus, 1 the difference between local download bandwidth and chunk source upload bandwidth was negative and 0 otherwise. Feedback is then sent to the tracker and the reputation score of chunk source is updated. [11] proposed modifying [12] by utilizing time decay function to give priority to recent behavior of peers. Both systems proposed in [11 and 12] provide a solution by augmenting the tracking service in BitTorrent. The solution did not provide means on how to share the collected information by the tracker about different peers with other trackers in order to make use of the collected information.

[9] proposed an autonomic framework for building trust in autonomic applications selected by user. The framework uses information exchange and communication between autonomic systems and their users as a tool to establish trust. Such approach reduces skepticism towards automation by increasing decision process transparency through direct experience of the user (trustor). The system does not store or share information developed about its users for the purpose of analysis or even future transactions with users.

In [17], the authors defined a number of trust computations for users' trust to choose among according to their preferences. The proposed system only utilized feedback as the only form of data. The proposed work only provides useful solutions given cooperation of nodes the important aspect, such as adhoc and sensor networks.

[44] introduced a systematic approach to trust establishment where trust can be established gradually through a mix of operations with user interaction and feedback learning. The trustee is software utilizing the autonomic computing concept for a dedicated system management function, termed Autonomic Manager (AM). The proposed system collects feedback about each execution of an AM and aggregates them for different executions across history to represent level of trust of system administrator in an AM. The proposed system provided no means for consulting experience of other system administrators, limiting the view of a trust to direct experience only.

[16] proposed personalizing Eigen trust model by weighting peers feedback according to level of trust in them according to trustor's perspective. The system did not consider the importance of time decay

function while aggregating data. [10] modeled personal trust using a graph representing direct experience of each trustor with the others as edges of the graph. Trust is calculated by averaging values of edges forming each path from trustor to trustee. The work presented an improvement to large scale market places, such as Amazon and eBay. However, the work did not provide means mitigating different forms of misbehaviors.

[6 and 20] proposes trust management systems for web services. In [6], the authors propose a system for automatic binding of web services in multi-agent systems. The system utilizes multi-agents to share information about services. The Agents calculate trust by collecting preferred QoS parameters from other trusted agents in the community and average them to produce the final trust values. In [20], a reputation management system for web services based on users' feedback about QoS parameters is proposed. In this work, each user selects among a set of QoS parameters according to his/her interest. Accordingly, trustor evaluates web services by providing feedback evaluating these parameters of web service. Trust is calculated according to user interest by aggregating feedback values for the selected parameters.

[27] proposed trust management systems for cloud computing. In [27], a trust management system for cloud based marketplaces is proposed. The system utilizes a questionnaire to provide feedback. Trust is calculated by weighing feedback by its sources credibility. Credibility is validated by comparing feedbacks of different sources together.

2.1.1.1.2. Hybrid Opinions Level 1

The following related works represent hybrid opinions level 1.

[4] proposed trust management systems for distributed file sharing application. In [4], the system used two parameters: the first evaluates file sharing service, while the second evaluates peers' participation in recommending good peers.

[15] proposed trust management framework. In [15], an autonomic trust management system for future internet architectures is proposed. The work defines future internet architecture as a group of autonomic nodes forming interconnected domains. The system used 2 types of trust managers, domain and inter-domain managers. Domain trust constitutes trust among parties of the same domain. Inter domain trust constitutes trust between parties belonging to different domains. Trust in a node is computed as the average of collected feedback within node's domain.

[19] utilized reputation management for routing in wireless ad-hoc. The proposed system collects direct and indirect observations about peers' behavior in packet routing. Trust is calculated using weighted average of most recent direct and indirect observations.

[25] proposed a trust management system based on multi-agents. The authors proposed to calculate trust via a third party SW agent. The calculated trust is a weighted average representing direct experience and indirect experience.

2.1.1.1.3. Hybrid Opinions Level N

The following related works represents hybrid opinions level N.

[5] proposed trust management system for distributed systems. In [5], the system utilized cookies to log peer's satisfaction in transaction with other peers. When querying trust, the logged satisfactions are retrieved from the stored cookies. If no available cookies were found, local cookies are used to find trusted peers who had experience with trustee.

[26] proposed a generic trust management framework. The authors proposed to represent the assessment of trust as a multi-class classification problem and use SVM as a solution. The proposed system enables collecting different forms of trust data and uses an algorithm to obtain the better suited to train a SVM. Accordingly, the trust level prediction model is obtained. The proposed system did not provide flexible means to choose other computation methods other than SVM.

2.1.1.2. Credential-based

The Credential-based class represents systems utilizing information about measures that qualify trustee for transaction. A trustee may be a service consumer, a service provider or a third party. These measures include Association, Safeguard and Certification. Association represents an official registration with a certain trusted group. Safeguards represent warranty or guarantees. For instance, refunding payment if the service outcome is not as agreed upon is a considered warranty while using money deposits may be a form of guarantee. Certification represents some kind of tests that should be passed by a trustee in order to measure and certify his/her capabilities in delivering certain outcomes.

2.1.1.2.1. Safeguard

An important system which basically is an association credential based is PayPal [36]. In the online auction systems for e.g. eBay [37], a considerable number of buyers utilize PayPal as a trusted third party.

Simply PayPal holds the purchase credits and does not deliver to buyer until the item(s) as described by the seller are received by the buyer. Only then, sellers are delivered their amount.

2.1.1.3. Popularity-based

The Popularity-based class represents systems utilizing information about how popular a trustee is. A popular trustee may be a service consumer, a service provider or a third party. A popular service consumer is one who is having transactions with many providers under a common subject of trust. For instance, a wireless communication client for T-Mobil, Verizon, U.S. Cellular and ATNT. A popular service provider is one who is having transactions with many clients, such as T-Mobil as a wireless communication service provider. A popular third party is one who is providing support for many transactions, such as PayPal [36]. Support may be in the form of guarantees, discovery services, trust service, etc.

Popularity of a trustee mainly falls into one of six categories: Cost, Clarity, QoS, Pervasiveness, Complexity and Hybrid. Popularity based on cost represents a trustee offering a certain cost resulting in being accepted or refused by relatively large number of trustors. For instance, a service provider who is offering challenging values for his services. Popularity based on clarity represents a trustee who provides certain amount of information about a given subject in order to reduce its vagueness. The information may result in the acceptance or refusal of trustee among relatively large number of trustors. For instance, consider delivering goods services of any e-marketplace. The delivering service may be popular and thus trusted among marketplace users, if it enables consumers to track their purchases starting from the warehouses and ending at their doors. In other words, the delivering service is being transparent or clear to users. Popularity based on QoS represents a trustee offering certain QoS levels which are considered accepted or refused among a relatively large number of trustors. For instance, Verizon as a phone carrier network provides the relatively good QoS and thus is popular. Popularity based on pervasiveness represents a trustee known among relatively large number of trustors and thus may be considered more trusted or less trusted. This knowledge emerges from the direct or the indirect experiences of trustors. For instance, if we consider a software solution sold by Microsoft. Microsoft is highly pervasive in software world and thus their solutions are generally trusted. Popularity based on complexity represents a trustee offering a certain subject at a certain degree of complexity. Such complexity results in the trustee being accepted or refused by relatively large number of trustors. For instance, if we consider two libraries offering a solution to a given problem and both solutions are reliable; however, the first library would be

more popular if it provides a simpler Application Program Interface (API) to users than the second. Finally, the hybrid popularity is considered a hybridization of the aforementioned popularities. For instance an internet service provider may be popular and thus trusted due to his relatively high QoS and low cost.

2.1.1.3.1. Popularity Based on Clarity

The work proposed in [9] represents popularity based on clarity. [9] measures trust in autonomic systems by the degree of transparency in their decision process. This is achieved by measuring the amount of information exchange and communication between autonomic systems and their users. Such approach reduces skepticism towards automation by increasing decision process transparency through direct experience of trustors.

2.1.1.3.2. Popularity Based on QoS

The work proposed in [1] illustrates popularity, based on QoS. [1] proposed trust management systems for distributed file sharing application. In [1] the system recognizes trust in a peer as the difference between its contributions and consumptions in the network.

2.1.1.4. Hybrid

Finally, the Hybrid class represents hybridization of two or more of the aforementioned systems.

2.1.1.4.1. Direct Experience and QoS Popularity

The following related works represent Hybrid Systems of Direct Experience and Provider Popularity.

[13] proposed a reputation management system for BitTorrent networks. The system evaluates peer's contribution as difference among upload bandwidth of peers. Peers choose to download according to their direct experiences.

[22] proposed trust management systems for cloud computing. In [22], the proposed work augments a cloud with a trust management system based on EigenTrust [4].

[21] addressed trust management in social networks and identified trustworthiness in this domain to be the weighted average of: the quality component of trustee, the evolution of trustee's trust rating over time and shifts which occur in trustee's behavior which is the derivative of trustee's quality component. The quality component of trustee is again the weighted average of the number of social relationships of

trustee, quality of the relationship with the trustor, and feedback from different participants in the network about trustee. The feedback is defined as the fraction of good votes cast for the trustee. The quality of trustee relationship a with trustor is modeled as a scoped random walk in which a random walker originates its walk at trustee and randomly follows the relationship links of trustee and the subsequent trustor at which it arrives up to a small number of steps.

[24] proposed a trust management system for internet of things. The proposed system utilizes weighted direct and indirect experiences and a set of parameters to mitigate maliciousness in the network. The parameters include honesty of service providers, cooperativeness of nodes in network and similarity of interests between trusting parties.

2.1.1.4.2. Direct Experience, Association and Pervasiveness

The work proposed in [18] represents Direct Experience, Association and Pervasiveness. In [18], a trust management framework is built up of 4 main components namely, content, communication, computation and counter action. The computation component computes trust based on direct, indirect experiences and group relationships information.

2.1.2. Classification of Trust Data

Trust computations illustrated in the previous classification can be applied to many forms of trust data. Figure 2.2 illustrates a classification of different forms of data that can be used to compute trust. In the figure, we may find that trust data can be classified into two main categories; namely, Input/Out data sent to/from the trust management system and data which describe the trust management system itself. The Input/output data includes different forms of data describing the four parties of a transaction; namely the service, the provider, the consumer and the broker. The trust management system data includes, function, structure and operation of the trust management system. The function trust data describes the overall operation of the system. This includes what trust computation and data it uses. The structure trust data describes the trust of the trust management system, such as a distributed p2p system or a centralized system…etc. Finally, operation trust data describes how the trust management system works including details of different operations performed.

If we go back to the Input/Out data form, the data about the three parties, the provider, the consumer and the broker, may fall into one of the four main categories; namely, behavior, diffusion, context and interests.

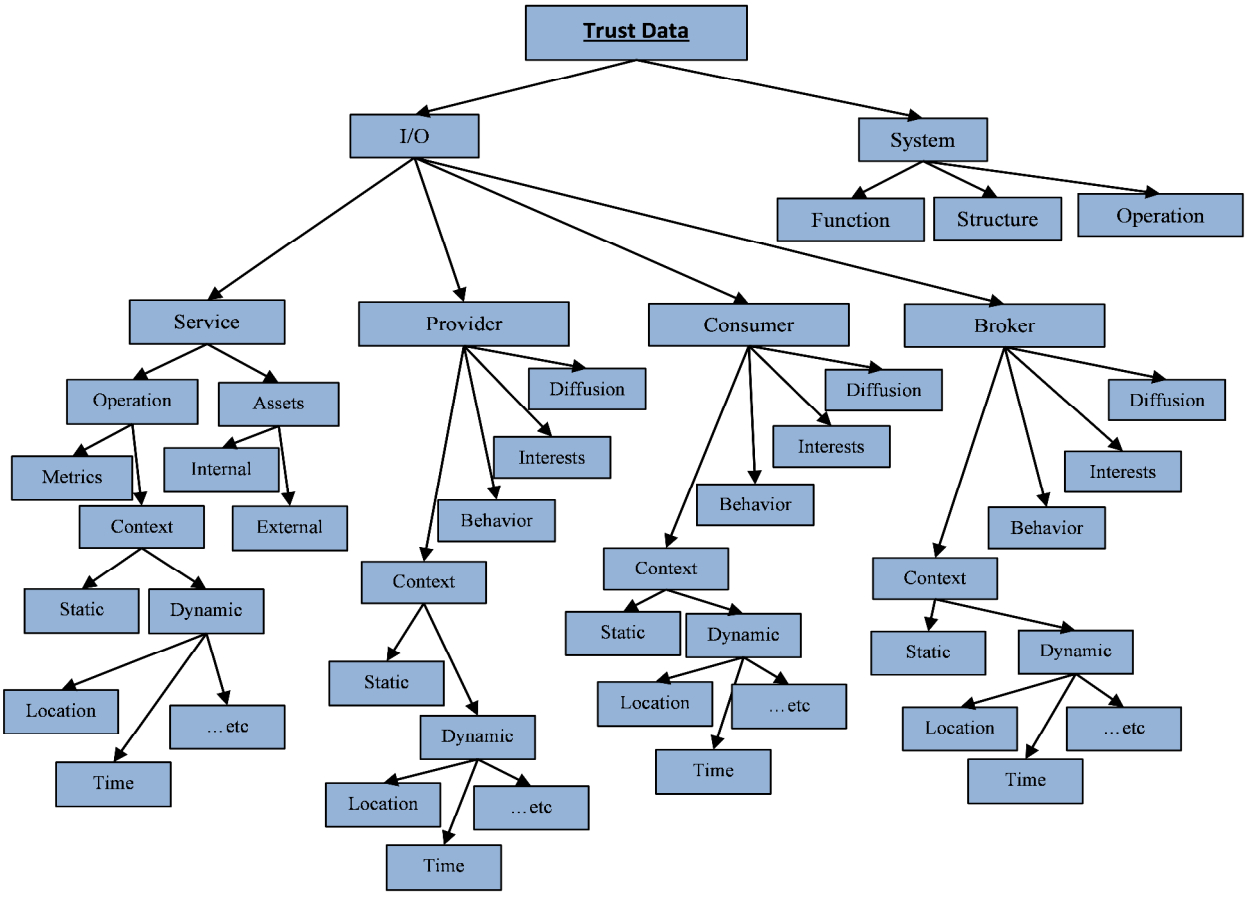


Figure 2.2 Different forms of trust data.

2.1.2.1. Party's Data

2.1.2.1.1. Behavior

The behavior data category includes all forms of information which describes party's exhibited behavior concerning a certain subject. For instance, [4, 7, 8 , 6, 10, 11, 12, 16, 21, 22 and 27] collected data about usefulness of feedback provider. [2, 3, 4 17, 20, 21, 22, 23, 24, 25, 27 and 36] evaluated service provider behavior by collecting feedback. [5] stored its evaluation of provider behavior in cookies.

2.1.2.1.2. Diffusion

The diffusion data category includes all forms of information which describes the popularity of a party. For instance, T-Mobile is a popular service provider. In literature, PayPal [37] is well known among eBay and internet communities as being a trusted third party for money deliver. Besides, in [1, 4, 13, 22 and

24] the systems recognized the more contributing and cooperative peers as more trusted than the consuming peers. [21] consider number of social relations as one of parameters evaluating trust, while [18] considers group relationship information as parameters for computing trust.

2.1.2.1.3. Context

The context data form incudes all forms of data which describes specific context according to which the party is being trusted in. In literature, [26] proposed a generic framework which included four main aspects according to which authors build their system's generic feature, among them most importantly was to enable trust calculation for different contexts. The system did not provide classification for different forms of context data. In general, we may classify context data into static and dynamic. Static context data form includes features of a party which does not frequently change, such provider being selfish, greedy…etc. Dynamic context data form may include time and location. Location data category includes all forms of information which describes a party's location. This may include geographical or geospatial locations. The geographical includes coordinates of a party as being in a continent, in a country, in a city. For instance, we consider a broker from Nigeria and another from Europe. The European may be more trusted due to his location data. Geospatial includes location in relevance to certain entity. For instance, in literature, [15] recognized future internet as a network of interconnected domains. Domain trust constituted trust among parties of the same domain, while inter domain trust constitutes trust between parties belonging to different domains. Here, to compute trust nodes geospatial location is classified as being from same or from another domain. Time context data form includes data which describes certain data or time information about party. For instance in November, internet service provider's availability gets low due to maintenance of servers.

2.1.2.1.4. Interest

The interests' data category includes all forms of information which describes specific subjects of interest of a party. In literature, [6 and 20] allow trustor to select his preferred QoS parameters which can be used in trust computation. [24] considered similarity of trustors' interests as one of the parameters computing trust.

2.1.2.2. Service

Now if we consider the service, we may find context, metrics, local resources used, and environmental resources. These types of information can be classified into two main categories; namely, operation and assets of the service.

2.1.2.2.1. Operation Context

[11 and 23] used time of invocation of service to provide the most recent observations more significance over the old observations.

2.1.2.2.2. Operation metrics

Here, [7, 8, 9, 15, 19, 20 and 26] collect data evaluating QoS outcome of services.

2.1.2.2.3. External Assets

[19 and 26] collected data about congestion in environment, representing environmental resources data form.

2.1.2.2.4. Local Assets

Up to our knowledge, no work in literature collects direct data value; however, the following works collect feedback evaluating users' satisfaction about local assets of a service. [9] Measured degree of transparency of autonomic service via amount of communication with its user. In this case, feedback evaluating degree of communication of autonomic service is collected, representing collected data about internal assets of communication of service. The works in [4, 6, 7, 8, 11, 12 and 20] collect feedback evaluating QoS delivered by service depicting form of data about internal assets, producing QoS.

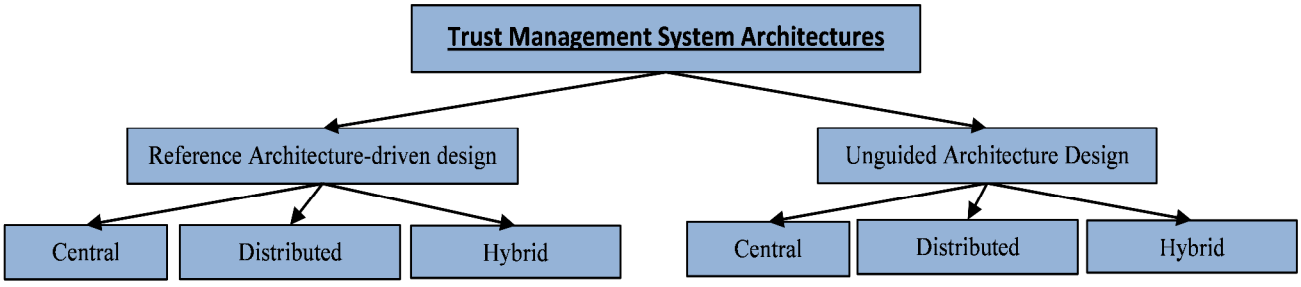


Figure 2.3 Different forms of trust architectures.

2.1.3. Classification of Trust Management System Architecture

In this section, we present a general classification of trust management systems' architectures. Figure 2.3 illustrates our classification. We begin our classification by identifying the main reference model of the architecture. This includes Reference Architecture-driven Design and Unguided Reference Architecture Design. The Reference Architecture-driven includes systems which utilize a framework or a reference architecture to design and build their systems. The Unguided Reference Architecture Design represents the systems which do not follow a reference architecture or a framework in their design. Next in the classification tree, the system's basic distribution model may fall into one of the following categories: centralized, distributed and hybrid architecture.

2.1.3.1. Reference Architecture-driven Design

The following represent systems which utilize Reference Architecture or framework model in their design.

2.1.3.1.1. Centralized Architectures

Centralized system architectures mainly operate one or more central server(s) where all trust management operations are performed. In literature, many systems exist; however [9] proposed an autonomic framework for building trust in autonomic applications selected by the user. The framework uses information exchange and communication between autonomic systems and their users as a tool to establish trust. Such approach reduces skepticism towards automation by increasing decision process transparency.

2.1.3.1.2. Distributed Architectures

There are many models which adopt distributed architecture, where different nodes have trust management operations and communicate with its counter parts to exchange data and operations.

[15] proposed trust management Framework. The work proposes an autonomic trust management system for future internet architectures. The work defines future internet architecture as a group of autonomic nodes forming interconnected domains. The system used 2 types of trust managers: domain and inters domain managers. Domain trust constitutes trust among parties of the same domain. Inter domain trust constitutes trust between parties belonging to different domains. Trust in a node is computed as the average of collected feedback within the node's domain.

In [18], a framework is proposed which is built up of 4 main components replicated across all nodes; namely, content, communication, computation and counter action. The content component describes the trust information elements. The communication component defines interaction among trustees to exchange trust data. The computation component computes trust. The counteraction component is responsible for informing other trustees of recent perception of trustors. The system only utilized fixed forms of trust computations and enabled users to extend the proposed components to build their systems.

[26] proposed a generic trust management framework and utilized it in their system design. The framework design includes four main aspects according to which authors build their system's generic feature, including, enable different forms of traffic inspected data, enable trust calculation for different contexts, use different levels of trust to support flexibility of trustor actions and use different entities (other than trustor) to collect trust data. The authors proposed to represent the assessment of trust as a multi-class classification problem and use SVM as a solution. The proposed system enables collecting different forms of trust data and uses an algorithm to obtain the better suited to train the SVM. Accordingly, the trust level prediction model is obtained.

However, the proposed system did not provide flexible means to choose other computation methods other than SVM. In addition, it did not consider uninspected forms of trusted data, such as warranty or guarantees.

2.1.3.2. Unguided Reference Architecture Design

The following represent systems which do not utilize Reference Architecture or framework model in their design.

2.1.3.2.1. Centralized Architectures

Many systems, in literature, adopt the centralized architecture with no reference for their design. They mainly operate one or more central server(s) where all trust management operations are performed. In literature, [1, 2, 3, 4, 7, 8, 11, 12, 15, 16, 17, 21, 22, 23, 24, 27 and 37] represent such systems.

2.1.3.2.2. Distributed

Distributed architecture includes the distribution of all operations across the network. In literature, [5] adopts this concept, where it proposed a system which stored node experiences locally and accordingly compute trust. When local experience is not sufficient, other nodes' experiences are utilized.

2.1.3.2.3. Hybrid

Hybrid system architecture exists where some operations of the system are performed at a central node, while other operations are performed in a distributed fashion. This includes in literature the distributed storage architectures in [25]. The authors proposed an immediate neighbor storage scheme for securing trust data and maintaining its integrity and confidentiality, while utilized a centralized third party agent which collects direct and indirect experiences from other agents in network.

[6, 13, 19 and 20] utilized distributed monitoring operations across the network while having a centralized node computing trust.

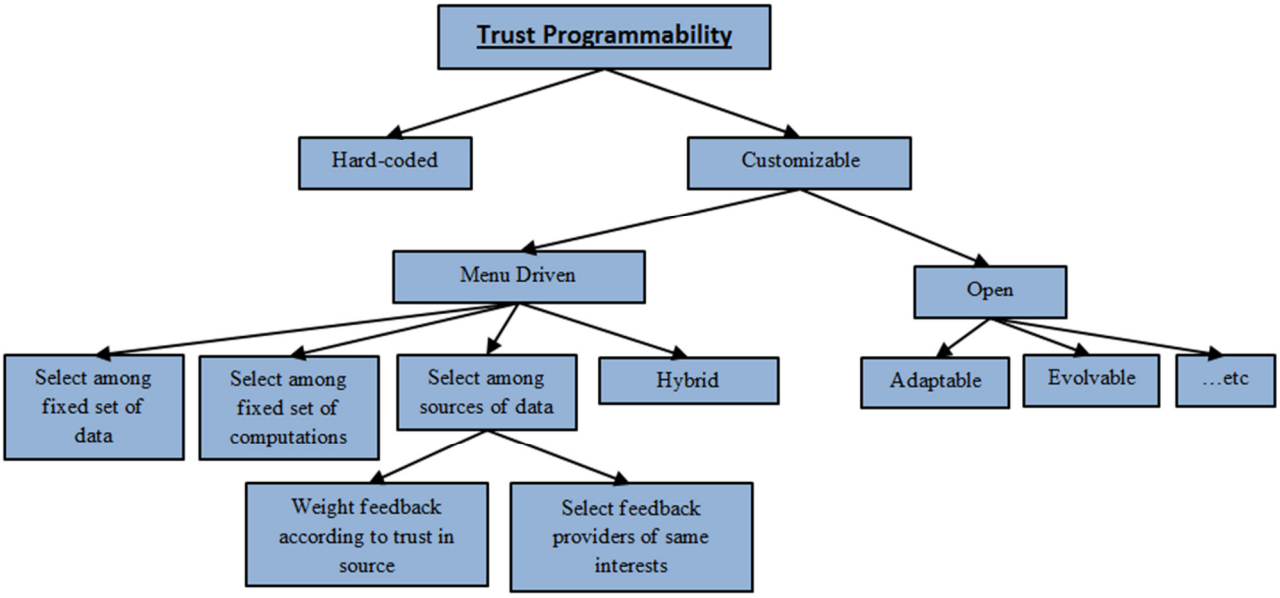


Figure 2.4 Different forms of programmability of trust.

2.1.4. Classification of Trust Programmability

There are three important aspects which differentiate trust management systems from one another; namely, trust computation, trust data and trust architecture. Personalization of trust implies considering the first two aspects, where a personalized system should include a wide variety of trust data and trust computations. Among these, the system selects which better suits the needs of the trustor.

In literature, some systems did provide some degree of personalization while other did not. In this section, we try to put a scope on these systems through a simple classification which presents the programmability of trust features embedded in these systems. Figure 2.4 illustrates our classification. In the figure, we classify systems programmability into two categories: hard-coded (un-personalized), customizable.

2.1.4.1. Hard-coded

In the hard-coded category, systems primarily operate on fixed data model and fixed set of trust computation(s). The data model may include one or more forms of trust data as discussed in figure 2.2 and may also include one or more trust computation(s) as discussed in figure 2.1. The term fixed here implies that the same data and trust computation(s) operate on all trustors in the same fashion regardless of any diversity in trust requirements. In literature, [1, 2, 3, 4, 5, 7, 8, 9, 11, 12, 13, 18, 19, 21, 22, 23, 25 and 37] represent this category.

2.1.4.2. Customizable

In this category, a trust management system can be customized such that trustors' requirements may be met. The degree of customization here is classified into two main categories; namely, menu driven and open.

2.1.4.2.1. Menu-driven

In the menu driven category, trustors can select among fixed set of options in order to satisfy his/her needs. These options include selection among fixed set of data, selection among fixed set of computations, selection among sources of data or a collection of the previous options.

Selection among fixed set of computations include the following [6, 15, 17 and 20].

Selection among sources of data includes two categories; namely, weighing feedback according to trust in the source and selecting feedback providers of the same interests. The first category includes the works of [10, 16 and 27], while the second includes the works of [24]. Besides, [9] provides this feature, where it proposed a framework which builds trust in autonomic applications using information exchange and communication between autonomic systems and their users. Such approach reduces skepticism towards automation by increasing decision process transparency through direct experience of user (trustor).

Additionally, a hybrid approach exists among these features. This includes [26] in literature. In this work, the system selects among a fixed set of data (network traffic inspected data) and compute trust by weighing feedback according to trust in data source.

2.1.4.2.2. Open

In the open category, trustor's requirements are first considered and then the system accordingly finds suitable trust data and computations to be used. This includes the system being able to adapt and/or evolve. To adapt implies modifying existing trust computation in the system in order to better suit trustors' requirements. To evolve implies first recognizing the need for a new form of data and/or trust computation and second being able to utilize the newly introduced.

2.2. Related Work Analysis

Analyzing the presented works, we may find the following common features, less common features and gaps in literature.

Common features include:

1. Trust is a one-way relationship, whereas consumers can evaluate providers while providers cannot evaluate consumers.
2. Ensure credibility of trust data, where trust data (feedback) are weighted by reliability, or trustworthiness of its source.
3. Centralized architectures are favored among trust management systems.
4. No reference architecture exists for trust management.

Less common features include:

1. Customize trust according to trustor's requirement. [7] provided such a feature by enabling trustors to choose QoS parameters which suit them. [8] Enabled users to only utilize raters of the same interest as the trustor. [11] Enabled users to consider only feedback from preselected set of peers.
2. Trust may vary from a context to another. [26] Defined trust as a context for which each entity, user, or system has different trust features, measures, and different contexts.
3. Popularity trust computation is rarely used by systems.

Gaps in literature include:

1. Hard-coded methodologies and computations: trust as well as trust-based decision support computations and methodologies are not personalized and are generally hard coded and non-reconfigurable.
2. Limited forms of trust Data and Computations: Contemporary trust/reputation systems primarily utilize feedback and direct or indirect experience, while other forms of trust data and trust computations are hardly ever utilized.
3. Oblivious to mutual trust requirements: most existing works focus on trust issues with service providers but neglect the need to also trust the consumer.
4. Tight coupling of trust management operations: no separation of concern among different trust management operations with the exception of [18]. Such coupling limits flexibility and customizability of the system, where it hinders the ability of the system to utilize different operations from existing applications, such as analysis, monitoring and data management.

2.3. Background

In this section, we discuss works that are related to our work. This includes BitTorrent file sharing protocol an Affinity propagation clustering algorithm.

2.3.1. Overview of BitTorrent

The cornerstone of success in BitTorrent [42] file-sharing applications resides in the principal component of coupling contribution and rewarding in the network. Each peer in the network only rewards those who provide it with good download rates, designated as tit-for-tat rule [31]. Thus, the more contributions made (uploads), the more rewards (downloads) a peer can get from the network and vice versa. Here, the contribution or rewarding commodities are the chunks of the file to be downloaded. In BitTorrent, the file to be downloaded is divided into smaller sized chunks designated as Pieces, typically 256k each. Each chunk is in turn split into smaller sized blocks of 16k. The blocks are transferred in numbers from/to peers in the network in the form of messages governed by each peer's Piece Selection and Choking algorithms, creating the mentioned rewarding/contribution scheme.

In order for the peer to start downloading a file, it has first to download a ". torrent" file which contains meta information about the target file, including number of pieces in the file, the hash value of each piece

and the tracker's IP address. The peer then contacts the tracker and registers itself in the swarm after receiving swarm setup information, including numbers of peers, each peer download achievement and peers' addresses. A Peer who does not have a complete copy of the file is known as a Leecher while the one who does is the Seeder. A Swarm is a group of peers engaged in series of upload/download activities of Pieces of a specified file. In a swarm, each Leecher is connected to a number of Seeders and Leechers. Among the Leechers and every 10 seconds, each peer selects a fixed number of peers (3, 4 or 5 peers, varies from an implementation to another) to which it responds to their Piece requests. These responded-to Leechers are known as un-choked peers, while the un-choking process is termed Peer Choking. These un-choked peers are selected according to the tit-for-tat rule. i.e. the top 3, 4 or 5 —according to implementation— Leechers who currently provided the best download rate service. Typically, current download rate is calculated according to a rolling average of the download rate during the past 20 seconds. Another un-choking operation is performed in parallel to Peer Chocking operation, namely, optimistic un-choking. Simply, every 30 seconds a peer selects a random Leecher for un-choking. The philosophy behind optimistically un-chocking a peer is to explore neighbors providing good services. Seeders adopt pretty much the same un-choke policy where they select a group of (3, 4 or 5 peers, usually varies from an implementation to another) Leechers to upload Pieces of the file. In old versions of Bit Torrent clients, Seeders would un-chock the fastest downloading Leechers from them. In more recent versions, seeders bandwidth capacity is distributed uniformly to connected Leechers. When un-choked Peers select Pieces for download and sends Piece request messages to peers un-choking it. Typically, pieces are selected according to their rarity in the swarm, where the more rare the Piece is, the higher priority for download. After successfully downloading a piece, the peer advertises its success via broadcast message, including piece number, termed as Have message. These Have messages provide peers with an insight about the distribution of number of copies in the swarm, thus enableing them to select rarest pieces. The process of selecting the rarest piece for download is termed as Rarest First Piece Selection.

2.3.2. Overview of Data Clustering

Data clustering is a very important, rather than an effective technique, for statistical data analysis and classification. In literature, we may find many applications, including machine learning, pattern recognition, image analysis, information retrieval, bioinformatics, data compression, and computer graphics. In our proposed trust management system, we utilize data clustering as a technique to reduce

size and complexity of trust data, thus enhance our system's scalability. This was achieved by utilizing a descriptive model of trust data rather than the raw trust data itself. The descriptive model, simply, is the output of the clustering operation. This includes the data of cluster heads and the classifications of all data points. More details can be found in section 4.2. In this section, we provide an overview of some of clustering techniques identifying their strengths and weaknesses from our work point of view.

Affinity Propagation [31] is a clustering algorithm that is based on message passing between clustering data points. The main strength of this clustering algorithm, from our point of view, is that it does not require the number of clusters in a data points set prior to the clustering operation. This point is crucial to our work, where the dynamics caused by different parties and the environment, make it impossible to prior identify the number of clusters. The algorithm rather calculates and exchanges two values among data points; namely, responsibility and availability, according to which it decides the number of cluster points and classification of each point to each cluster. The responsibility reflects the accumulated evidence for how well-suited a point is to serve as the exemplar for a given point taking into account other potential exemplars. While the availability reflects the accumulated evidence for how appropriate it would be for a point to choose another point as its exemplar, taking into account the support from other points that this point should be an exemplar. The algorithm performs a number of iterations till reaching a steady point where either the cluster boundaries remain unchanged over a number of iterations or reaching a predetermined number of iterations. Within each iteration, the algorithm updates availability and responsibility values of each data point.

One of the drawbacks of Affinity Propagation clustering algorithm is its complexity, where it is well known by its cubic order of time complexity.

***k*-means clustering** [53] is a form of an unsupervised machine learning technique for data partitioning. Simply, the operation starts by providing number of clusters (k) according to which n data points are to be partitioned. The next steps are then performed:

1. Select random k points representing initial centroids.
2. Assign each data point to the group that has the closest centroid.
3. Recalculate the positions of all centroids.

4. Repeat Steps 2 and 3 until the centroids no longer move.

The main strengths of *k*-means reside in its simplicity and the low complexity O(n), compared to other clustering algorithms. However, we should mention that *k*-means clustering is also significantly sensitive to initial selected centroids. Additionally, there is no general solution to finding optimal number of clusters for a give data set. A simple solution is to compare the results of multiple runs with different *k* values and accordingly choose the best.

***k*-medoids clustering** [54], similar to *k*-means clustering, the procedure starts by providing the number of clusters *k* as required. The next steps are then performed:

1. Select random *k* points from the data set to serve as medoids.
2. Repeat:
    a. Assign each point to the most similar medoid.
    b. Random select a non-medoid point.
    c. For each medoid, compute the cost of swapping with the selected non-medoid
    d. If the cost is less than 0, then swap

    Until no change

Similar to *k*-means, *k*-medoids is simple in implementation; however, similarity calculations is performed in each step resulting in computationally expensive similarity operations. Most importantly is its initial requirement of the number of clusters same as *k*-means.

Density Based Spatial Clustering of Applications with Noise (DBSCAN) [55] is a density based clustering technique. This type of clustering is classified as a partitional type, where for a give set of points, the high dense regions are classified as clusters, while low dense regions are considered noise. Clusters are defined according to two input values; radius and minimum number of points. The steps for performing DBSCAN are as follows:

1. Select a random point r among a given data set.
2. Find all data points that are within the given radius of point r such that satisfy minimum number of points.

3. If point *r* has more than the specified minimum number of points; then a cluster including all these points is formed.
4. If *r* has fewer than the minimum number of points, within its radius, and is in the neighborhood of a core point, then go to next point in data set.
5. Continue the process until all of the points have been processed.

One of the main advantages of DBSCAN is its time and space complexities which is $O(n^2)$. However in some cases where the radius value is chosen in a meaningful way, its results $O(nlogn)$ time complexity. Additionally, DBSCAN does not require the number of clusters a priori as in the case of *k-means* and *k-medoids*. However, DBSCAN cannot cluster data sets well with large differences in densities, since the radius and minimum number of points combination cannot then be chosen appropriately for all clusters. Additionally if the data set and their scale are not well understood, then selecting a meaningful radius and minimum number of points values can be difficult.

If we are considering DBSCAN as an alternative for Affinity Propagation clustering, then we might gain the advantage of reduced time and space complexities. However, we may face issues in classification when dealing with non discrete values of metrics. Non discrete values are expected to generate differences in densities within the same cluster resulting in classification error. A solution to this problem might be in formalizing the required metrics values into discrete metrics. However, a key challenge here is not to lose the advantage of the reduced complexity in metrics conversion process.

We may face another issue when dealing with DBSCAN in our system where, we may receive unexpected behavior values, due to noise or oscillation in the environment, resulting in difficulties in radius and minimum number of points selection. A solution to this problem might be in the utilization of machine learning, where the accumulated knowledge would assist in choosing the appropriate radius and minimum number of points values.

2.4. Conclusion

In this chapter, we provided an overview of related works in literature identifying their strengths and weaknesses. We also presented some of the technologies involved and used in our research, which included BitTorrent file sharing protocol and Affinity Propagation clustering technique. Additionally, we briefly surveyed number of clustering techniques identifying their strength and weakness and their suitability to our work. We presented trust management systems in literature using four aspects, including

trust computation, trust data, trust architecture and trust personalization. For each of the four dimensions, we provided a classification of related works. Based on our classification, we identified common features, less common features and gaps in literature. Gaps included:

1. Hard-coded methodologies and computations: trust as well as trust-based decision support computations and methodologies are not personalized and are generally hard coded and non-reconfigurable.
2. Limited forms of trust Data and Computations: Contemporary trust/reputation systems primarily utilize feedback and direct or indirect experience, while other forms of trust data and trust computations are hardly ever utilized.
3. Oblivious to mutual trust requirements most existing works focus on trust issues with service providers but neglect the need to also trust the consumer.
4. Tight coupling of trust management operations: no separation of concern among different trust management operations with the exception of [18]. Such coupling limits flexibility and customizability.

Chapter 3

3. Trust Management System Architecture

In this chapter we present the proposed trust management reference architecture. We first provide a formal description of our model illustrating its main components and different entities involved, using Extended Backus-Naur notation. Then, we illustrate the operational environment where an implementation of the reference architecture can be used identifying trust interaction among different roles of different parties in the environment as well as trust management system role and components. Then, we illustrate components of the reference architecture including each structure, sub-structure, function, operation and personalization factors. We finally illustrate our design principal upon which we built our reference architecture then contrast its' features against that of other works in literature.

3.1. Formal Model

We define our trust management model using Extended Backus–Naur form (EBNF) [40] notation. Our model comprises a set of service providers, service consumers, brokers, services and a set of trust management services. We may describe model structure accordingly.

<TMS-model> = <providers>, <consumers>,<Brokers>, <services>, <TMServices>;
<providers> = {Provider-Id};
<consumers>= {Consumer-Id};
<brokers> = {Broker-Id};
<service> = {<service>};
<TMServices>= {<TM-Service>};

A service is a set of actions to be performed by a service provider on one or more resources of his/her own, for the welfare of the service receiver. A resource is whatever is needed to provide a service. Accordingly, we model a service as follows:

<service> = {<type>};
<type> = {<type>} | <composition>;
<composition> = {<type>}|{<instance>};
<instance> ={Provider-Id ,Instance-Id} ;

Service type refers to different functionality of a given service. Service composition indicates a group of services forming a composite service. Service instance indicates a picture of the service. While service instance pair indicates the service provider hosting or outsourcing the service. For more clarification, let us consider the file sharing service illustrated in figure 3.1. We have different service types as booking, file sharing, online banking…etc. For File sharing service, we may have types as Nutella and BitTorrent. In BitTorrent, we have different types of services, including seeding leeching and tracking. For tracking, we have two types, centralized and distributed. In centralized tracking, we have two types tracking provided by Vuze or µtorrent programs. In Vuze, we have tracking instance for Dark Knight and Spiderman. Finally, for each instance, we may find a provider.

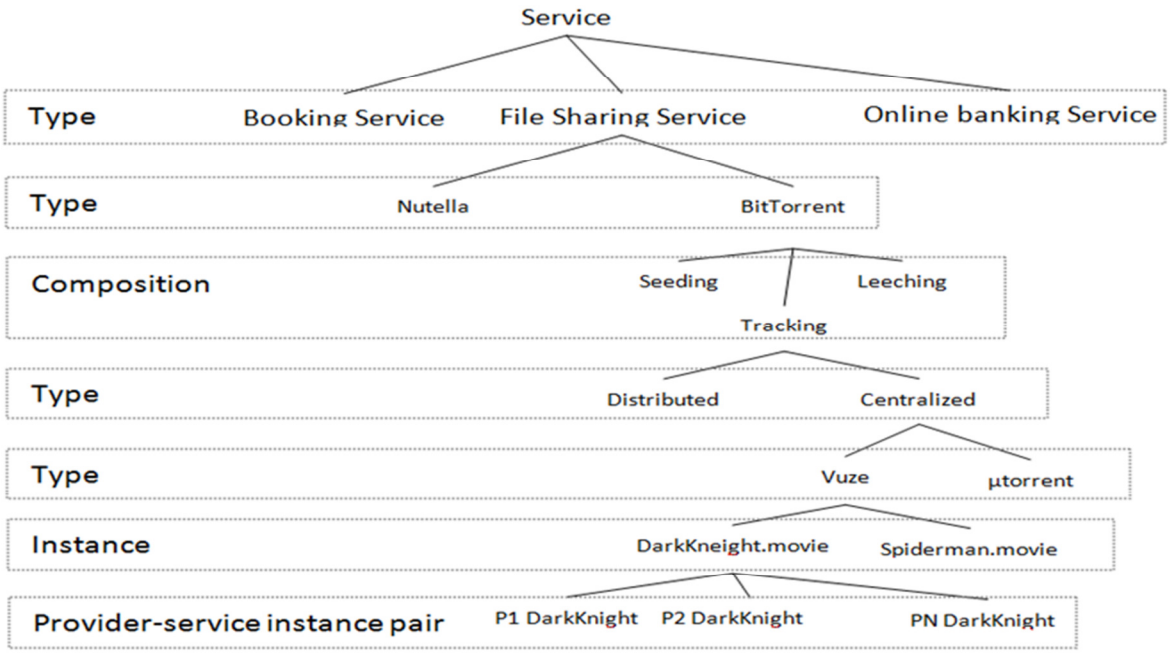


Figure 3.1 Service description example.

A service provider is the party that hosts or outsources the service instance in order for service requestors to invoke it. A service consumer is the party which seeks, requests and invokes service instances. Here, party denotes a human, a computer program or an autonomous agent.

Consumers have no prior knowledge or guarantees about providers. Hence, the need to establish trust prior transacting. Service providers place certain regulation for their intended consumers in order to

maintain their own good reputation. For instance some providers get disrepute by consumers who provide false feedback about them despite their satisfaction with the provided service. Hence, mutual trust is needed prior to transacting. Accordingly, in our model, we enabled consumers and providers to separately define their trust requirements. These requirements may differ from one party to another given the diversity in needs, trust profiles and capabilities. The collection of trust requirements for a single party is termed trust preferences. These preferences are mapped onto trust profile of the intended transaction party resulting in a successful transaction.

A Trust Profile represents the set of properties which characterize a party, while Trust Preferences are the set of requirements needed by a party to trust another party for transacting. We describe Trust Preferences and Trust Profiles in our model as follows:

<TrustPreferences> = {<Record_group>},<owner> | {<Record_group>}{<record>},<owner> |
{<record>}<owner>;
<Record_group> = {<record>},<owner>;
<record> = <Quality-metric>[,<owner_threshold>];
<Quality-metric> = metric-name;
<owner_threshold> = value;
<owner> = provider-Id | consumer-Id;
<Trust-Profile>={<Quality-metric>, value},<owner>;
<Quality-metric>= <computation-method>, name;
<computation-method>=method-id,{<monitored-parameters>},return-type;
<monitored-parameter>= name, value;

As illustrated, a record is composed of a Quality-metric and owner's threshold for the Quality metric value. The record represents the party's trust requirement regarding a certain property in the intended transaction party.

On the other hand, a trust Profile constitutes a set of quality metric and value pairs and the profile owner. A Quality-metric has an identifier and is computed by a computation method, while a computation method has an identifier, a set of parameters which are used to compute its value and type of this value.

A personalized trust management system, primary role is evaluating different parties in order to select which satisfies users' trust preferences.

Trust preferences of a party can be identical or similar to some other party's preferences. Similarity implies 2 types of records, an identical and different. The identical may be similar to other party's preferences, resulting in a hierarchal tree-like-structure showing a heritage tree of preferences among

parties. We used Record_group tag to enable inheritance from other parties in our description to the model. For better illustration, let us consider the following example.

Consider 2 organizations which are trying to help research institutes in selecting the suitable cloud providers. The first organization is formed and funded by a group of industrial companies (OIC) and the second organization is formed by the state of VA government (OVA). The goal of OIC is to help research institutes across the states to find quality cloud providers which can serve reliable SW, platforms and infrastructures. While OVA is trying to help research institutes in VA find local cloud providers suitable for their research requirements. Both organizations are trying to help research institutes by providing them with trust preferences document which directs them to best cloud providers in the market who can achieve their goals. Let us first consider the set of requirements defined by OVA.

Req#1: provider work in VA. (Meet state quality requirements).

Req#2: provider outsources services from volunteers in VA.

Req#3: at least 3 offices in VA.

Req#4: average services fees $30-$70 per month.

The main philosophy of OVA as inferred from the requirements is to direct research institutes to local cloud providers who are licensed in VA. The main idea is that the law in VA can protect research institutes when cloud providers do not abide by the contract. We may observe the recommendation for local providers from req#1, req#2 and req#3. Besides, the state tries to direct research institutes to prices in local market as inferred from req#4.

Now, let us consider the set of requirements defined by the OIC.

Req#1: resources owned by provider from any of the founders: Microsoft, T-Mobile, Fujitsu or CENets.

Req#2: cloud provider has a National license.

Req#3: bandwidth provided not less than 20Mbps.

Req#4: minimum number of servers 5.

As we can see from the defined requirements, the OIC is targeting good performance.

Now, if we consider 3 research institutes, the first (user 1) does research work in VA and the second (user 2) works on high performance computing and the third (user 3) is in VA and works on high performance computing. The first user may simply inherit from the document provided by OVA and add its personal requirements (e.g. BW not less than 18Mbps). The second user (user 2) targets high performance, hence he/she would inherit preferences document as provided by investment bureau. The third user (user 3)

needs to work in VA targets high performance, so he/she would simply inherit both preferences. Figure 3.2 and 3.3 illustrates that.

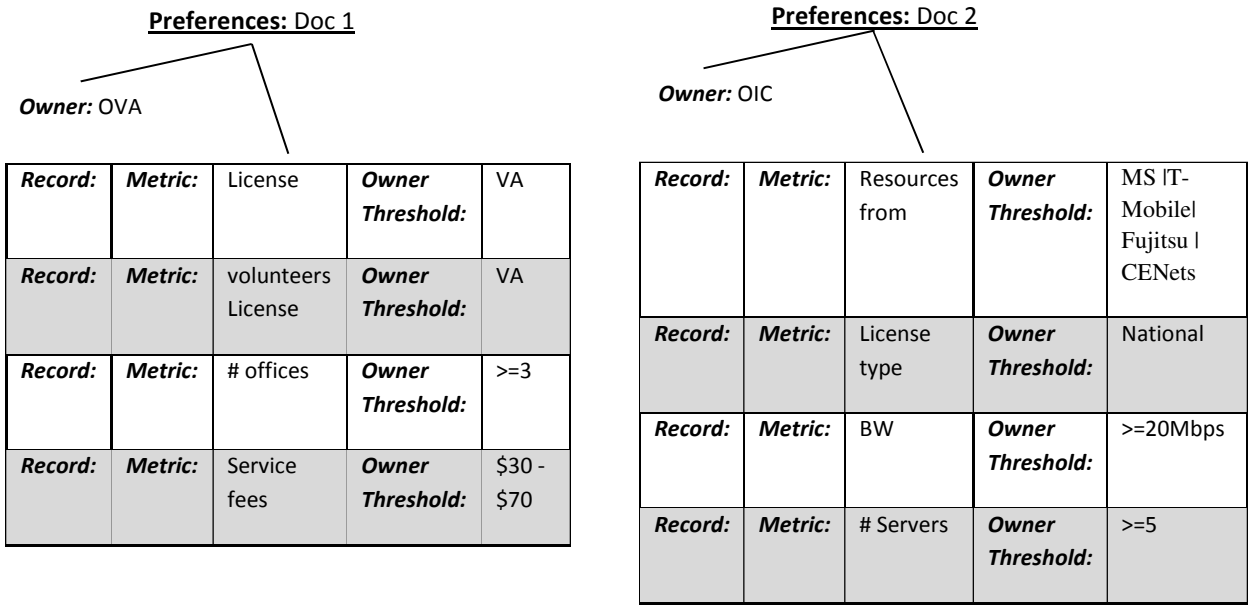


Figure 3.2 Illustrates structure of trust preferences document provided by OVA and OIC.

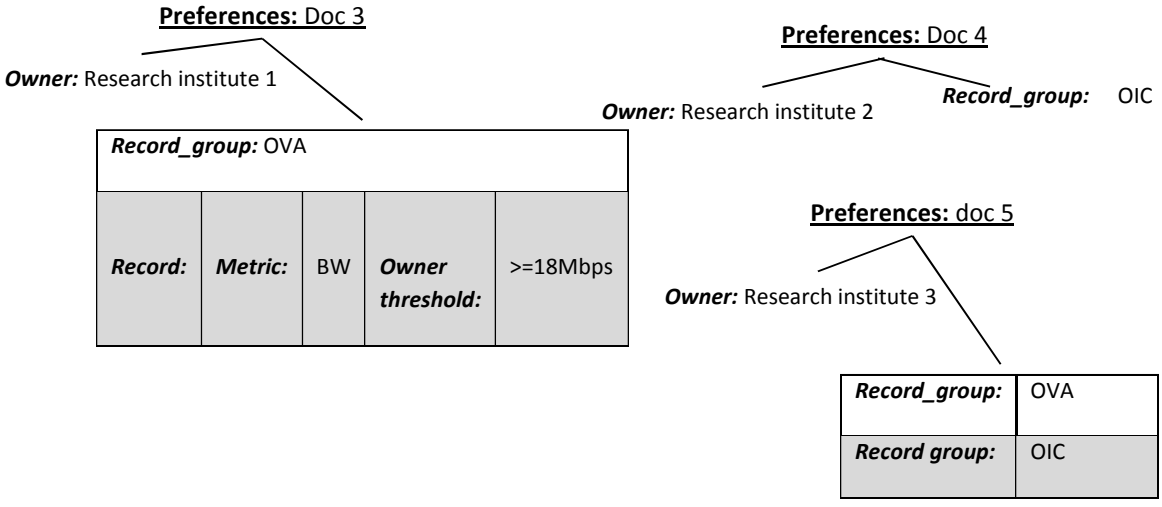


Figure 3.3 User 1, user 2 and user 3 preferences document structure.

Trust management takes the form of a service hosted and consumed by different parties. If we consider the very beginning of a market at time zero, consumers and providers at that time start searching for ground zero trust; namely, an ultimate trust management service and a provider. By ultimate, we mean trusted by default. The default trust here is based upon specific features which characterize the trust service and the trust service provider. Such as a trusted third party as the government. The ultimate trust management service is used by a party to select a suitable trust management service from many alternatives in the market. Recommending the ultimate and the preferred trust services are done by the broker role. A broker is a service provider who considers requirements and properties of a party and accordingly recommends transaction parties who can satisfy these requirements. At time zero, the broker recommends an ultimate trust management service and provider while in other times a broker recommends other services and providers, as well.

In our model we define trust as the belief or disbelief that another party, for a said subject of trust in a given context, has the ability to exhibit a set of acceptable actions in the future, for the welfare of the trusting party. Accordingly, we may describe trust as follows:

<trust> = <belief> | <disbelief>,<trustor>,<trustee>

<belief> = <Intent>,<Integrity>,<Capability>,<Results>,<SoT>,<Context>

<disbelief> = <Intent>,<Integrity>,<Capability>,<Results>,<SoT>,<Context>

<trustor>=provider| consumer

<trustee>= provider| consumer

Our trust management reference architecture decouples trust management operations into five operations: Decision Management (DS), Expectation Management (EM), Analytics (AN), Data Management (DM) and Monitoring Management (MM). Each of the five operations is a service by itself. Trust management services existing in the model, either follow or do not follow our reference architecture. Accordingly, we may describe trust management services as follows:

<TMService>=<RATM-based>|<non-RATM>;

<non-RATM> = <trust-management>;

<trust-management>= <service>;

<RATM-based> = <RATM-composition>;

<RATM-composition>= {<DS>}, {<EM>}, {<AN>}, {<DM>}, {<MM>};

<DS> = <service>;

<EM> = <service>;

<AN> = <service>;

<DM> = <service>;

<MM> = <service>;

From our description, we may observe that each RATM-based trust management service can be hosted and provided by a separate node in the network. Additionally, similar services can federate to form an overlay network for expectation, analytics, monitoring and data management. More details about function, structure and operation of each RATM-based trust management service can be found in the next section.

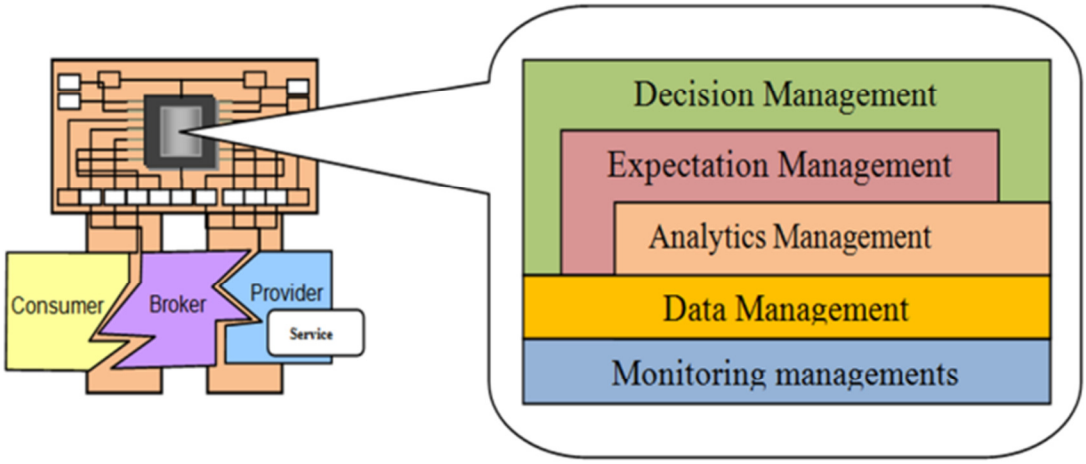


Figure 3.4 RATM reference architecture.

3.2. Reference Architecture

In this section, we provide an overview of the proposed Architecture. Then, we present its services (components) and describe the structure, function and operation of each.

3.2.1.Overall Architecture

The proposed architecture operates in a service-based environment, where users of different roles (consumer, broker and provider) are using trust-driven transactions on services and resources to satisfy their trust requirements. A trust requirement is the set of requirements needed by a party to trust another party. It may include sets of metrics and the corresponding threshold values for each metric. For instance,

a video conferencing user on a highly-congested network may need a video stream application of 98% reliability and 95% availability.

For each trust requirement, there exists at least one set of capabilities and trust profiles which can satisfy it. By a set of capabilities, we mean all the assets that are used to deliver a service, such as software code, network bandwidth. By trust profile we mean the set of quality metrics values representing the expected outcome of a service. For example, the popularity of a service provider, where the more popular the service provider is, the higher the cost of him cheating or delivering unaccepted outcomes.

We may visualize the matching of trust requirements of a role (consumer, provider or broker) to capability and trust profile pair of another role, as the matching of the saw tooth of a lock and its corresponding key (figure 3.4). In the figure, the proposed architecture brings different roles together in a transaction by performing the aforementioned matching process. Technically, this is achieved using 5 customizable reconfigurable components responsible for the management of decision, expectation, analytics, data management and monitoring operations. These five components coordinate, communicate and exchange data to perform trust management operations.

The decision component operates with expectation, analytics and data management components to make intelligent decisions. The analytics, data management and monitoring are continuously operating while decision and expectation components request their operations on demand. The decision and the expectation components, both may utilize any generic component performing: analysis, data management and monitoring. In this way, the proposed architecture can be used to add to any application intelligent-customized trust management decisions and expectations, given the application's generic analysis, data management and monitoring, or may even add and define new analysis, data management and monitoring operations as needed.

The expectation component enables the role of trustor (consumer, provider or broker) to expresses his trust requirements: choose quality metrics evaluating the role of trustee (consumer, provider or broker), define thresholds values for the chosen metrics and preferred decision operations. Given these requirements, expectation management component utilizes analyzers and data management components to produce trust reports evaluating trustees. The decision component then makes decisions accordingly. Decisions include selection, rehabilitation, blocking…etc.

Finally, we need to note that our reference architecture design has taken into consideration that each layer is aware of its adjacent layer. In other words, we do no adhere to strict layering architecture.

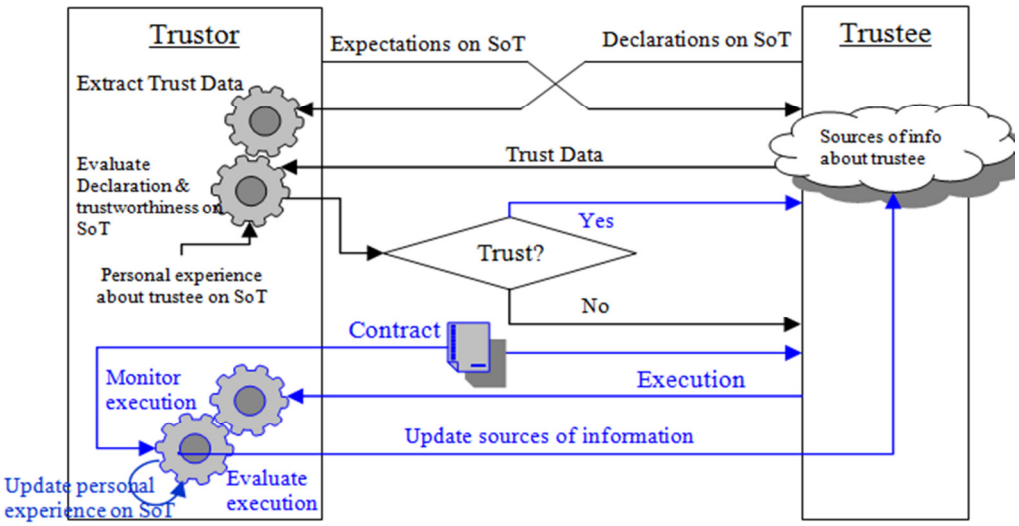


Figure 3.5 Trust establishment protocol.

3.2.2. Interaction

Among two roles of authenticated parties, for a given subject, the architecture applies the trust establishment protocol illustrated in figure 3.5. In the figure, a Trustor and a Trustee exchange expectations and declarations on the subject of trust (SoT). The trustor extracts needed trust parameters from trustee's declarations and then collects needed trust data from its own and others experiences. The trustor then evaluates trustee's declarations and accordingly develops the trust profile necessary for evaluation. In case the decision is to trust, the trustee is informed and a contract abiding trustee's provisioning is developed. During provisioning, the trustor monitors trustee's performance and accordingly evaluates trustee. Finally, the trustor updates collected trust parameters locally (personal experience) and remotely (sources of information about trustee).

3.3. Architecture Services (Components)

The five services (components) of the proposed architecture (figure 3.4) are Decision Management (DS), Expectation Management (EM), Analytics Management (AN), Data Management (DM) and Monitoring Management (MM). The functions, the structure, sub-structure, the operations and personalization factors of each service are described in the following sections.

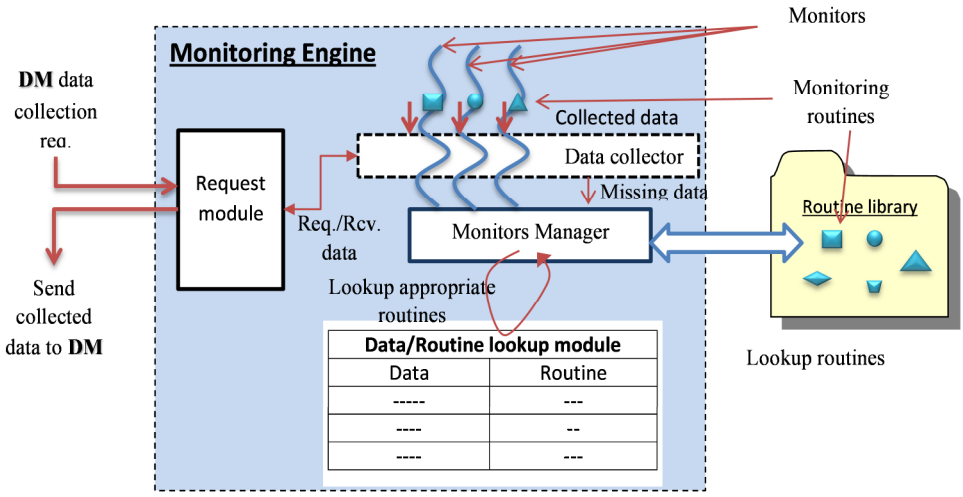


Figure 3.6 Monitoring management structure.

3.3.1. Monitoring Management (MM)

Function: responsible for capturing and collecting the required trust data.

Structure: illustrated in figure 3.6, includes:

1. Monitoring Engine: provides the environment necessary for creating, destroying and running monitors. It includes the following sub- components:
   i. ***Request module:*** receives requests from DM to collect data and replies to DM with the collected data
   ii. ***Monitors Manager:*** provides the environment necessary for monitors to run routines for collecting required data
   iii. ***Data collector:*** captures monitored data to be stored at DM and informs monitors manager of required data to be collected
   iv. ***Data/routine lookup module:*** lookup module necessary for selecting appropriate routines for each form of trust data.
2. Routine Library: holds routines for collecting different forms of trust data. The library is open for system administrator to add new decision procedures.

Operation: Monitoring Engine receives data collection request from DM:

DM informs Monitoring Engine of the required data to be recorded about a given role of a given party in an upcoming transaction. The Monitoring Engine accordingly selects suitable routines for data collection and starts them within the next transaction. On transaction termination, the collected data are forwarded to DM to be stored.

Monitoring engine sends collected data to DM:

Monitoring engine continuously manages the operations of monitors which continuously collect trust data. After a monitored transaction terminates, the Monitor(s) involved forwards collected data to the Monitoring Engine. The Monitoring Engine then forwards it to DM for storage.

Personalization factors

According to our presentation, in the following points, we may summarize the personalization factors of monitoring management component:

- Ability to collect any form of data evaluating transaction parties: the routine library is open for defining necessary monitoring routine(s) to collect any required data.
- Ability to collect data about different contexts: monitoring routines may be added to monitor changes in context. For instance monitoring location of a node or monitor congestion in a network.
- Ability to utilize different monitoring procedures: monitoring routine alternatives may exist and selections are made according to requirements.

3.3.2. Data Management (DM)

Function: is responsible for implementing storage, retrieval and replication policies necessary to manage system and trust data. In addition to maintaining: integrity, availability and privacy of system and trust data. Moreover, it is responsible for formatting heterogeneous data into a single accessible format for other components of the architecture to use.

Structure: illustrated in figure 3.7, includes:

1. Data Observer: ensures the existence of required trust data and manages the updates and query operations to Database. It includes the following sub-components:

i. ***Request module:*** receives data queries from DS, EM and AN and replies back with result sets. It further receives recoded data from MM to be stored.

ii. ***Query parser:*** parses data queries received by the Request module.

iii. ***Management engine:*** is the heart of DM where it manages data format exchange, identifies missing data in queries received and requests them from MM, stores collected data after replicating them at Reliability manager.

2. Data Homogenizer: includes routines necessary to change the heterogeneous format of the collected trust data, into a unified homogeneous format accessable by architecture components. It includes the following sub-components:

   i. ***Format exchange manager:*** receives format exchange request and manages format exchange operation.

   ii. ***Lookup module for data exchange routines:*** format exchange manager uses it to identify routines necessary for format exchange operations

   iii. ***Format exchange routine library:*** includes format exchange routines

3. Data Reliability Manager: includes a map for locating data and their replicas across the network. In addition it maintains integrity, availability, privacy and security of data. It includes the following sub-components:

   i. ***Data synchronizer and locator:*** receive requests for data storage and retrieval and manages their storage locations, including their replication

   ii. ***Lookup modules for data replicas' locations:*** is a lookup module necessary for allocating data replicas across the network.

4. Database: holds trust data.

Operation: Data Observer receives data queries from: DS, EM or AN or new collected data by MM.

- On receiving data queries, Data Observer uses Replication Manager to locate Trust Database and then executes necessary queries to trust database:
  - In case data was found, result set is created and forwarded back.

— In case no data was found, Data Observer informs a query source of the no data status, and requests MM of the missing data.

- On receiving Notification from MM about newly collected data.
  - Data Observer forwards collected data to Data Homogenizer to unify its format.
  - Data observer then forwards the unified data to Data Reliability Manager for storage.

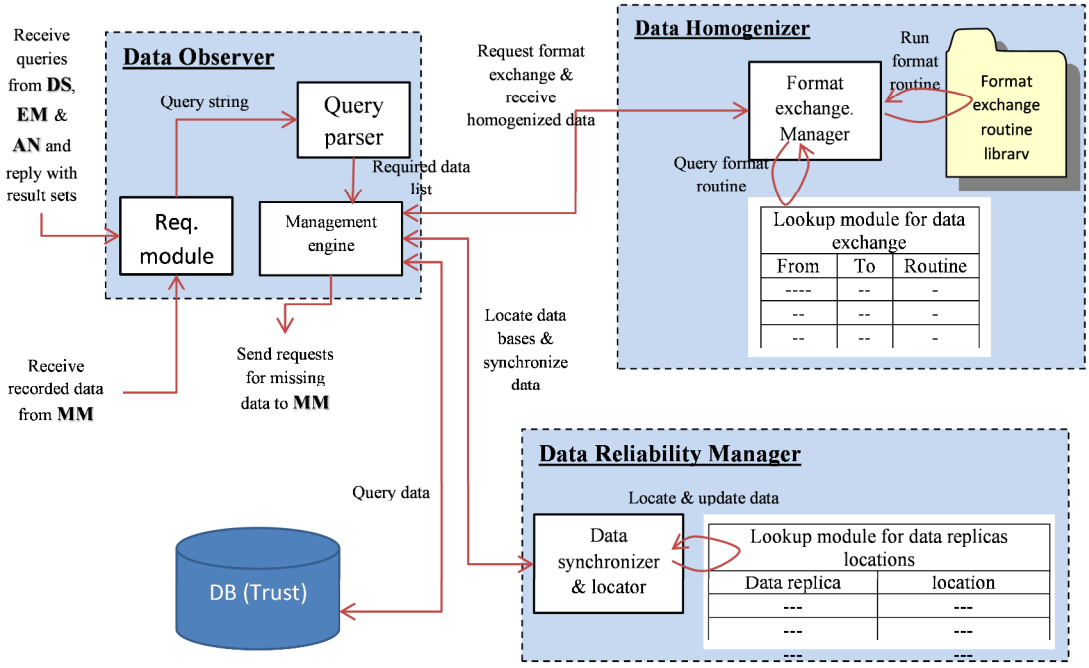


Figure 3.7 Data management structure.

Personalization factors

According to our presentation, in the following points, we may summarize the personalization factors of Data management component:

— Ability to store different forms of data: all data to be stored at the data management component is homogenized into a single accessable format for other components of the architecture to use.

- Ability to use different storage policies: this includes different algorithms for replicating data and storing them at different nodes/locations. The replication manager can be configured in the system to implement the required storage policies for securing data.

3.3.3. Analytics Management (AN)

Function: responsible for analyzing behavior of different roles of different transaction parties.

Structure: illustrated in figure 3.8 includes:

1. Analysis Query Manager: accepts queries from DS and EM and collects required data from analysis operations. It includes the following sub-components:

    i. ***Request module:*** receives requests for analysis from Ds and EM and replies back with analysis results.

    ii. ***Query parse:*** parses requests received and provides a list of analysis operations required for results generator.

    iii. ***Results generator:*** requests analysis results from operation manager and sends them back in appropriate formats to request module.

2. Operation Manager: manages continuous operation of analysis and collects analysis results for storage at DM. It includes the following sub-components:

    i. ***Analysis manager:*** provides necessary environment for running analysis operations and manages their continuous running states of operations. It is also responsible for adding new analysis operations from routine library.

    ii. ***Data collector:*** responsible for capturing analysis results and sends these results to DM for storage.

3. Data Selector: sends queries to collect parameters values from DM. It includes the following sub-components:

    i. ***Data query engine:*** prepares necessary data for analysis operations from DM.

ii. ***Analysis lookup module:*** is used by data query engine to identify necessary data for each analysis routine.

4. Analysis Library: includes a set of operations for analyzing behaviors of different roles of different transaction parties. The Library is open for system administrator to define new analysis operations for new trust computations.

Operation: The Operation Manager continuously performs all analysis operations. It communicates with DM through data selector to prepare all required parameters for different analysis operations. The operation manager also collects output of each analysis operation and stores it in DM.

The Analysis Query Manager receives queries from DS and EM and translates them to the results extracted from output of different analysis operations. These results are then forwarded to a requestor.

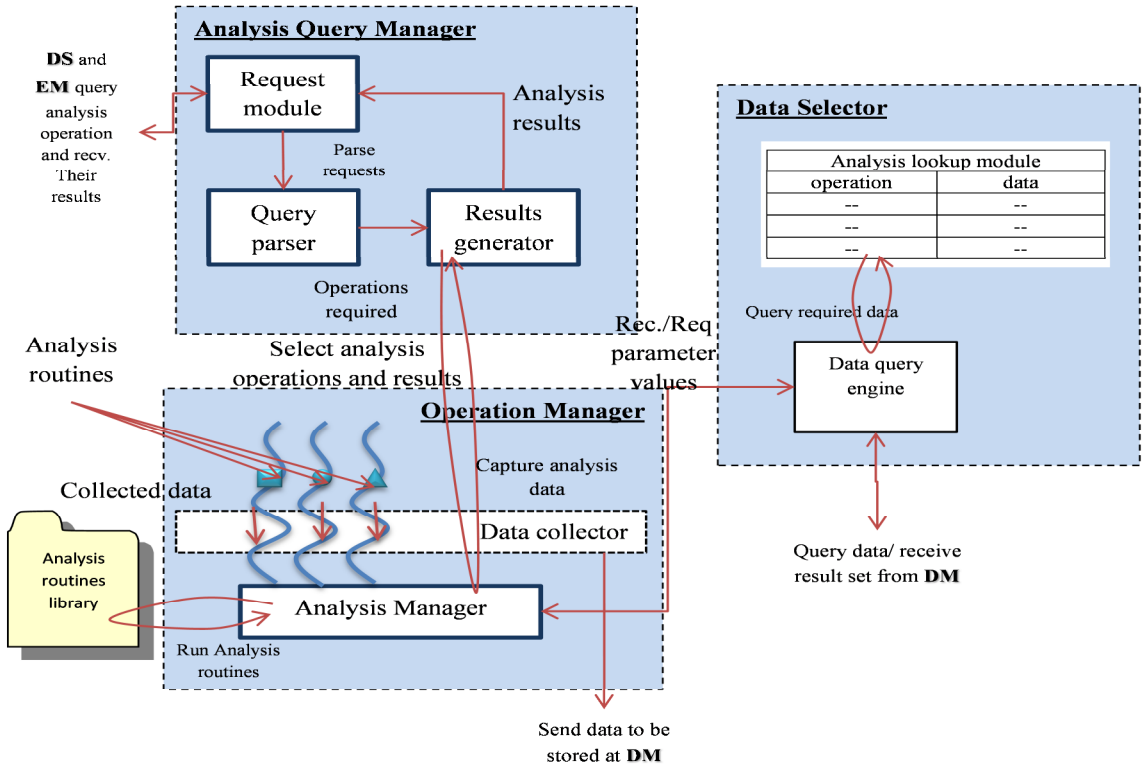


Figure 3.8 Analytics management structure.

Personalization factors

According to our presentation, in the following points, we may summarize the personalization factors of analytics management component:

- Ability to utilize different analysis algorithms: the analysis library is open to define all forms of analysis operations required by the system. The analysis operation may be implemented in a centralized or distributed fashion. i.e. implementer on a single or multiple nodes in order to enhance performance.
- Ability to utilize different forms of data: different forms of data homogenized by DM are utilized by analytics.
- Ability to perform analysis operations locally or remotely: users trust may utilize his local analysis component or switching it off and utilize an alternative on remote node(s).

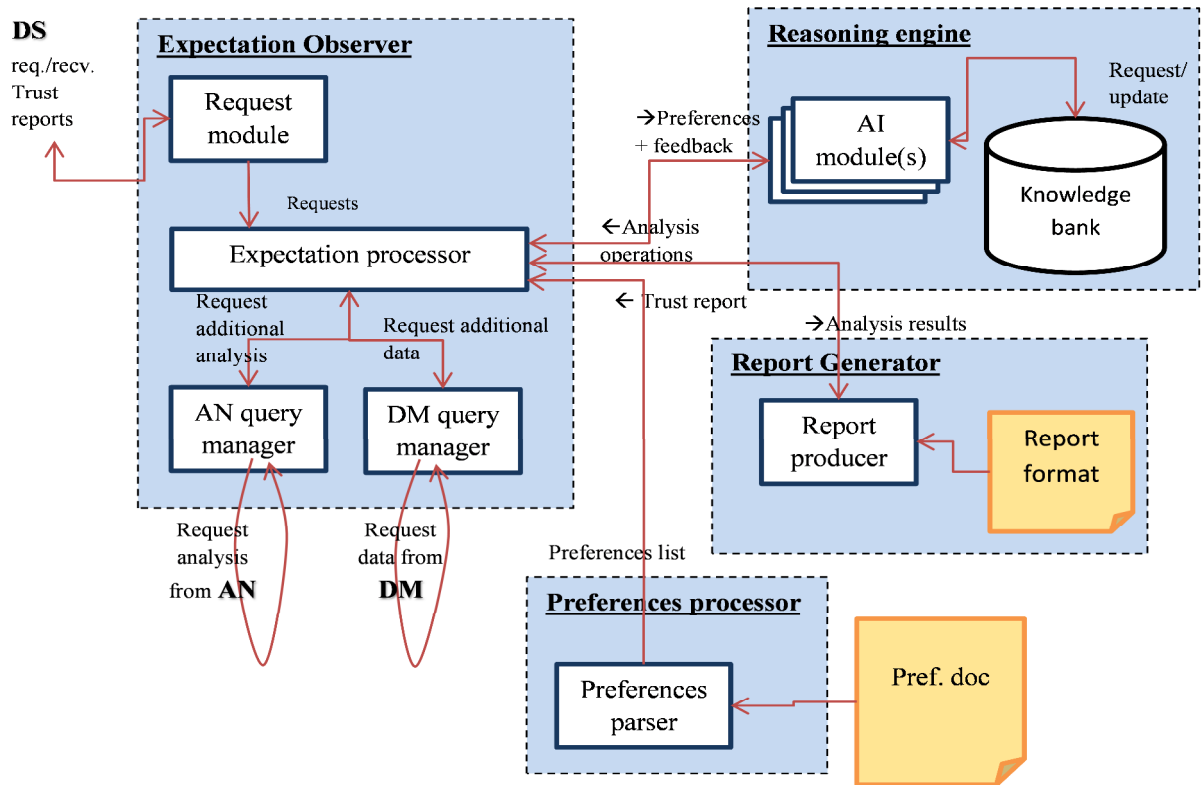


Figure 3.9 Expectation management structure.

3.3.4. Expectation Management (EM)

Function: is responsible for developing an expectation of the future behavior of different roles of different transaction parties given a defined set of trust requirements.

Structure: illustrated in figure 3.9 includes:

1. Expectation Observer: manages flow of execution to produce Trust Reports. A Trust Report is a set of trust profiles for a given role for a number of parties. For instance, the trust report about multimedia services in a network includes trust profiles for each available service alternative. It includes the following sub-components:

    i. ***Request module:*** accepts trust report request from DS and replies back with trust reports.

    ii. ***Expectation processor:*** is the heart of EM where it receives requests for trust reports from DS along with trustor's preferences from preferences processor. It requires necessary analysis and data from AN and DS and passes them to report generator which later on produces the required trust report.

    iii. ***AN query manager:*** manages queries sent and received to/from AN.

    iv. ***DM query manager:*** manages queries sent and received to/from DM.

2. Preferences processor: extracts trustor preferences, including preferred quality metrics, user's thresholds for quality metrics, preferred time window and preferred trust computation method. It includes the following sub-component:

    i. ***Preferences parse:*** it parses preferences document and provides a list of preferences to be processed.

3. Engine Reasoning: develops knowledge about analysis operations and accordingly selects the most suitable. It includes the following sub-components:

    i. ***AI modules:*** includes machine learning modules utilized by EM for reasoning and learning

    ii. ***Knowledge bank:*** data base holding the knowledge developed by learning operations.

4. Report Generator: receives analysis results and creates trust reports accordingly. It includes the following sub-components:

   i. ***Report producer:*** utilizes report format, trust data and analysis results to generate trust report.

   ii. ***Report format:*** includes the formatting for trust report

Operation: The Expectation Observer receives requests for producing Trust Report from DS. Accordingly, it extracts trustor preferences from Preferences Processor and passes preferences along with any additionally required data from DM to the Reasoning Engine to select suitable analysis operations(s). The Expectation Observer then contacts AN for analysis and receives results. Then, it passes the results to the Report Generator which may query DM for additional data. Finally, the Expectation Observer receives the trust report and then forwards it to DS.

Personalization factors

According to our presentation, in the following points, we may summarize the personalization factors of expectation management component:

- Ability to utilize different evaluation metrics: a user trust is enabled to define his trust preferences stating one or more preferred evaluation metrics and assigning threshold values to these metrics. Expectation management component utilizes appropriate analyses methods (from AN) to compute values for the preferred metrics.
- Ability to utilize different evaluation methods: the component is open to utilize any evaluation method defined in Analytics component or may even utilize any external analytics service from existing application.
- Ability to establish trust within a given context user trust may also define contexts within which he may establish trust and vice versa.

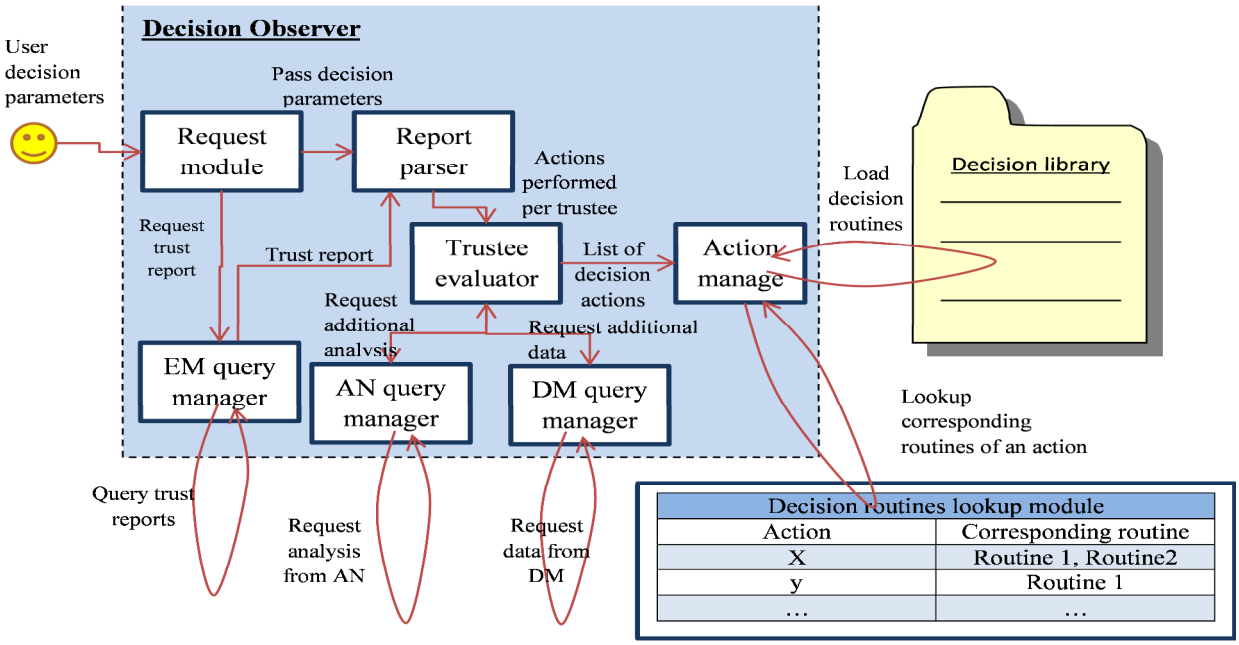


Figure 3.10 Decision management structure.

3.3.5. Decision Management (DS)

Function: is responsible for implementing trust-based decisions of a trustor. Such implementation includes the selection of interaction parties, feedback provisioning, type of penalties applied to misbehaving parties, duration of rehabilitation period for misbehaving parties and type of rewards for well behaving parties.

Structure: illustrated in figure 3.10 includes:

5. Decision Observer: interprets Trust Report received from EM, using AN and DM, into decision actions. It includes the following sub-components:

    i. ***Request module:*** accepts trustor's decision request along with necessary decision parameters. It passes these parameters to report parser and informs EM query manager of trustor decision request

ii. ***Report parse:*** parses trust report and produces list of actions which were observed about each trustee and hands it over to trustee evaluator.

iii. ***Trustee evaluator:*** considers the list of actions and requests additional data and analysis to trustees in list. It then identifies a list of actions to be taken regarding each trustee and passes that to action manager.

iv. ***Action manager:*** uses the lookup table to run corresponding routines per action.

v. ***EM query manager:*** manages trust report requests sent and received to/from EM.

v. ***AN query manager:*** manages queries sent and received to/from AN.

vi. ***DM query manager:*** manages queries sent and received to/from DM.

1. Decision Library: holds a set of decision routines.

2. Lookup Table: holds a list of behavior and corresponding routines signature.

Operation: The Decision Observer receives trust report from EM. Then, it identifies the required actions by checking for recorded penalties, rewards and rehabilitations for each role of a party in the given trust report. Additional analysis (AN) applied to additional data (DM) is performed in order to identify the required actions. Then, the Decision observer accordingly updates the list of the available roles for transaction and runs selection routine along with all necessary actions.

Personalization factors

According to our presentation, in the following points, we may summarize the personalization factors of decision management component:

- Ability to utilize different decision algorithms: the decision management lookup table and library are open for adding any necessary decision making algorithm which best suits user trust preferences. Additional analysis and data may be performed/collected, if needed, to assist the decision making process. This enables user's trust to have the following options:
    - Utilize different rehabilitation periods and processes.
    - Apply different rewarding and punishments
    - Utilize different rehabilitation periods
    - Utilize different threshold values

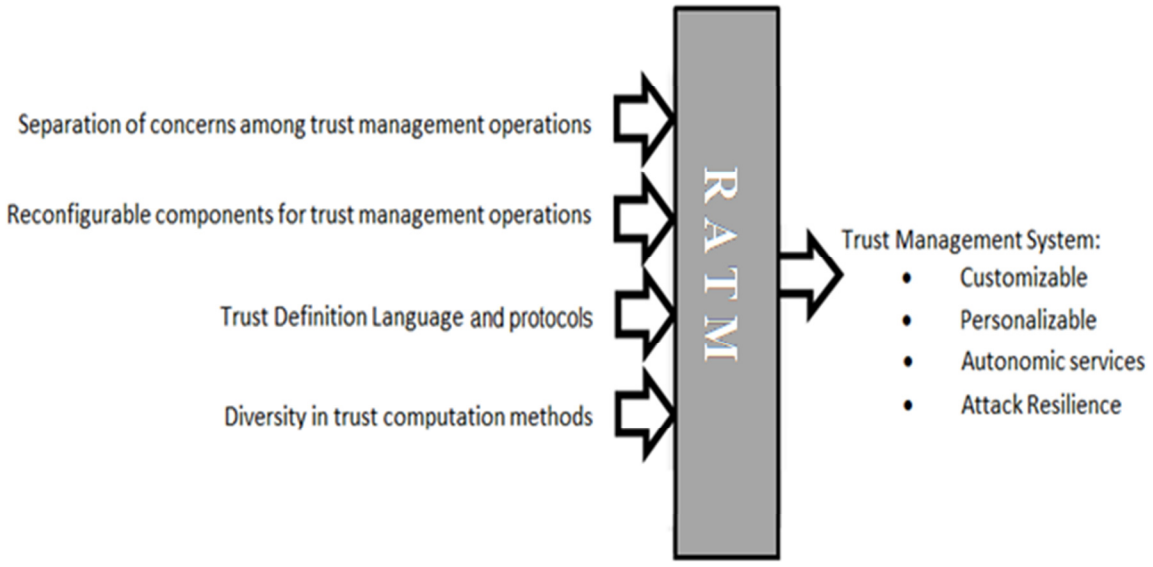


Figure 3.11 Reference architecture design principles.

3.4. Design Principles

The underpinning pillars upon which RATM is being built are illustrated in figure 3.11 along with the main features characterizing it.

They are as follows,

- Separation of concerns among trust management operations; trust management operations are performed each by a separate component, namely: Decision, Expectation, Analytics, Monitoring and Data management (discussed in next sections). Each component interacts and exchanges data with the other components to perform the overall operations of trust management. Such separation of concerns added flexibility to the architecture, where it enables the utilization of different operations from existing applications, such as analysis, monitoring and data management.
- In order to support flexibility and enable personalization, we added the reconfigurablity feature to each of the five components of the architecture. In this way, our architecture can be customized and reconfigured to virtually implement any trust management system.

- The reconfigurablity of each component is performed using TDL. Such language would enable trustors to define their requirements according to their needs. Additionally, it allows system administrators to modify or even introduce new evaluation methods which best suits users of the system. The reconfigurablity feature was based on careful examination of trust management systems in literature where we found that some systems are different from each other among their:
  - Trust computations (analytics) only. For instance, [2] and [23]. [2] used a weighed average scheme to calculate the trust value, while [23] builds a directed graph and uses it to represent the trusted and unknown trust or untrusted.
  - Forms of data (association and popularity along with experience as in [18]),
  - Methods of storage (e.g. cookies as in [5]),
  - Evaluations of the trustee (e.g. the difference between peers upload contributions in the network instead of average feedback as in [13]),
  - Decisions (e.g. [23] decides to trust or un trust, while [11] defines 3 levels trust, no trust and unknown).
- Such reconfigurablity and flexibility enrich the proposed architecture and open room for having diversities of trust computation methods as well as trust computation method evaluations. We adopt the model of a service-based operation, where trust management operations are considered as services and may accordingly be evaluated using one of the trust computations defined in the system.

The above features collectively resulted in customizable and personalizable architecture which provides autonomic and attack resilient services of trust management.

Table (3.1 and 3.2) compares features of the five services of RATM-based system (PTMS) against that of the related works.

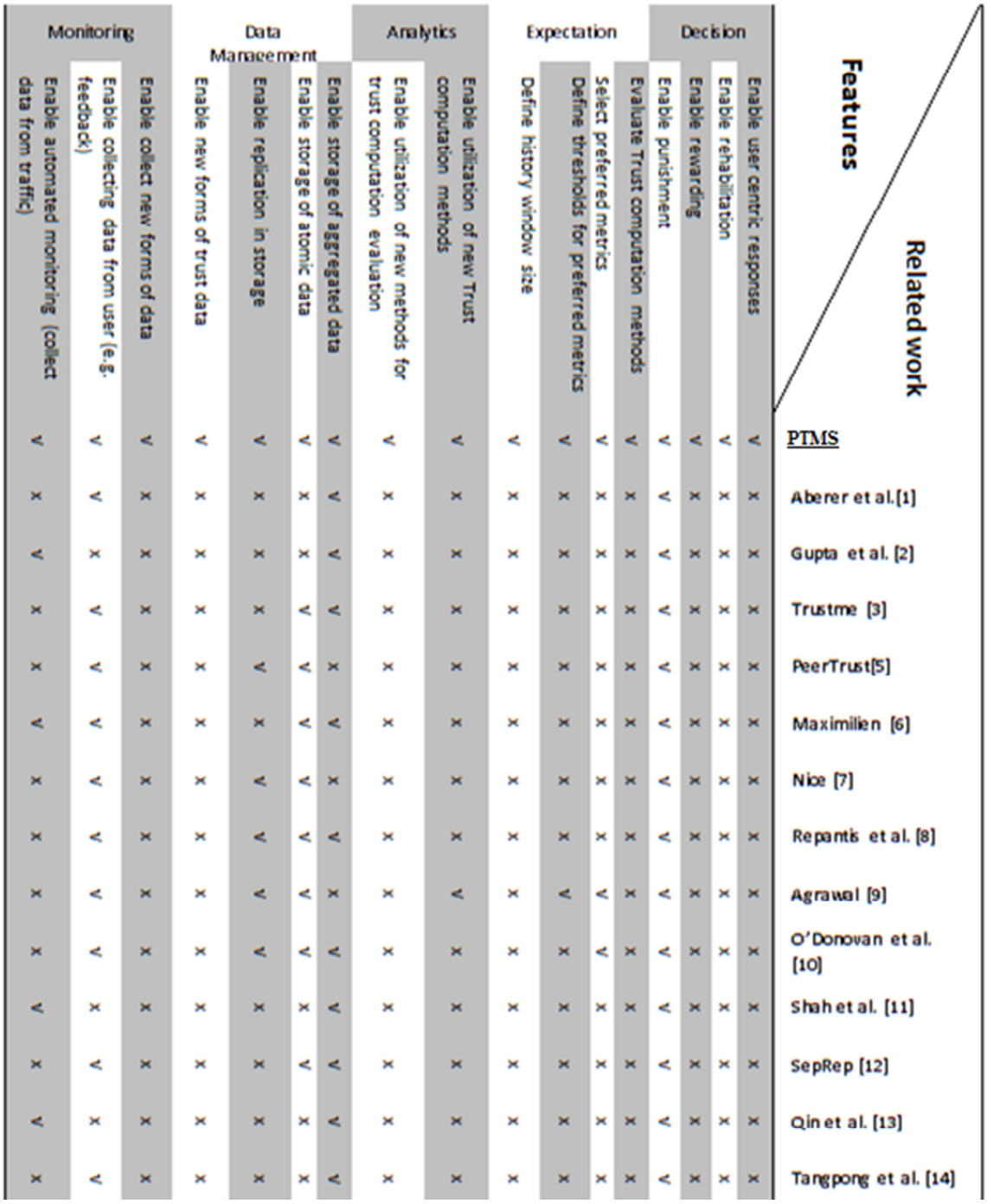
3.5. Conclusion

In this chapter, we present our proposed trust management reference architecture. Basically, we presented its working environment and identified different roles in this environment including: the service, the consumer the provider and the broker. We also illustrated the role of trust management system in such environment and identified the protocol followed in order to establish trust. Next, we presented the five, components building RATM: Decision, Expectation, Analytics, Data Management and Monitoring. Then, we discussed each component's function, structure, operation and personalization factors. After

presenting RATM, we finally highlighted the design principals upon which we built RATM and then contrasted features provided by RATM against that of works in literature. Our design principles include:

- Separation of concerns among trust management operations
- Reconfigurable trust management operations.
- Enabling users to describe their trust requirements, trust data and trust computations.
- Utilizing diversities of trust computation methods as well as trust computation method evaluations.

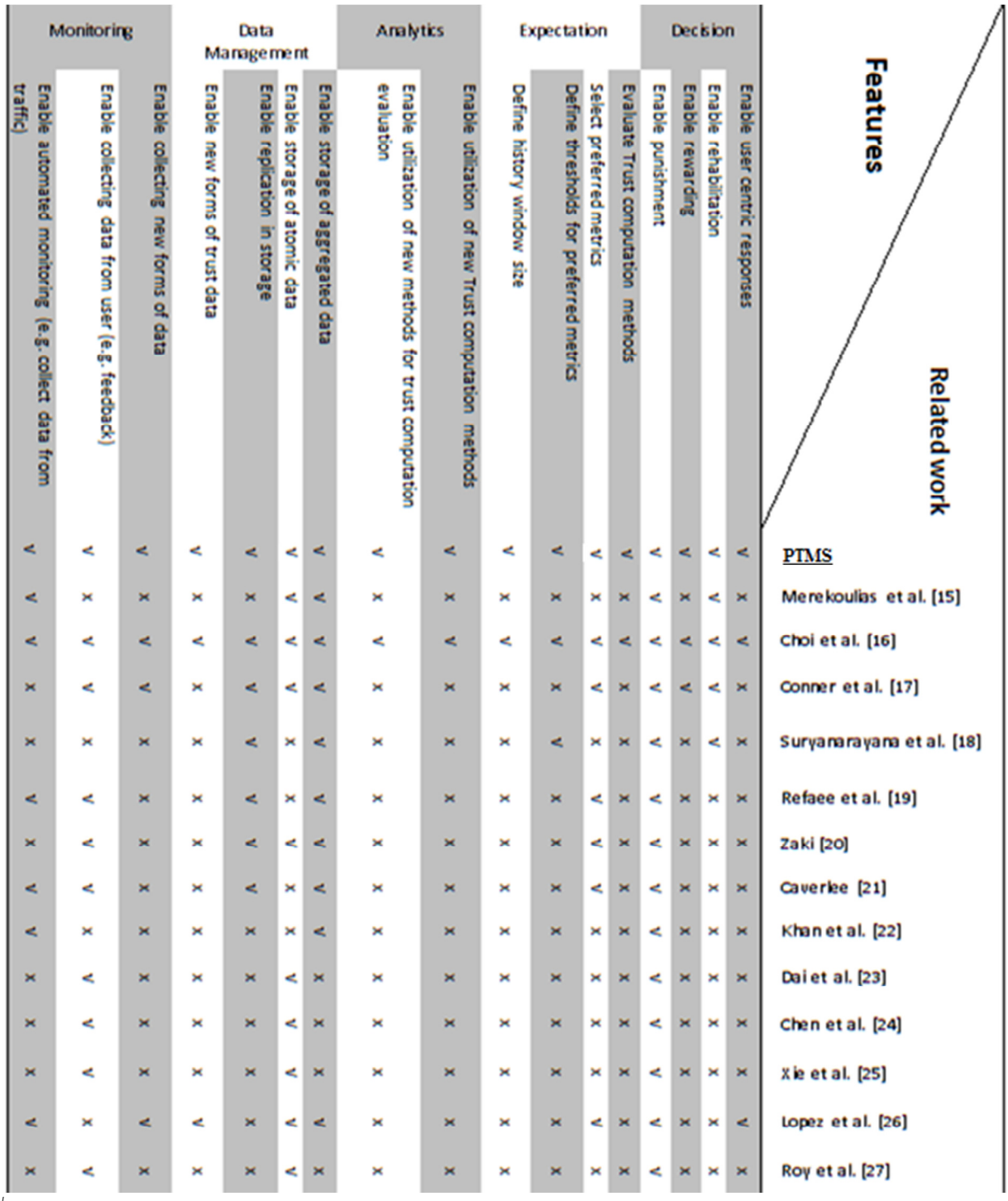
Table 3.1 Features of RATM compared to related work.



Features / Related work	Decision				Expectation			Analytics		Data Management			Monitoring		
	Enable user centric responses	Enable rehabilitation	Enable rewarding	Enable punishment	Evaluate Trust computation methods	Select preferred metrics	Define thresholds for preferred metrics	Define history window size	Enable utilization of new Trust computation methods	Enable utilization of new methods for trust computation evaluation	Enable storage of aggregated data	Enable storage of atomic data	Enable replication in storage	Enable new forms of trust data	Enable collect new forms of data	Enable collecting data from user (e.g. feedback)	Enable automated monitoring (collect data from traffic)
PTMS	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√
Aberer et al.[1]	X	X	X	√	√	X	X	X	X	X	√	√	X	X	X	√	X
Gupta et al. [2]	X	X	X	√	√	X	X	X	X	X	√	X	X	X	X	X	√
Trustme [3]	X	X	X	√	√	X	X	X	X	X	√	√	X	X	X	√	X
PeerTrust[5]	X	X	X	√	√	X	X	X	X	X	X	√	√	X	X	√	X
Maximilien [6]	X	X	X	√	√	X	X	X	X	X	√	√	X	X	X	√	√
Nice [7]	X	X	X	√	√	X	X	X	X	X	X	√	√	X	X	√	X
Repantis et al. [8]	X	X	X	√	√	X	X	X	X	X	√	√	√	X	X	√	X
Agrawal [9]	X	X	X	√	√	√	√	X	√	X	X	√	√	X	X	√	X
O'Donovan et al. [10]	X	X	X	√	√	√	X	X	X	X	√	√	√	X	X	√	X
Shah et al. [11]	X	X	X	√	√	X	X	X	X	X	√	X	X	X	X	X	√
SepRep [12]	X	X	X	√	√	X	X	X	X	X	√	√	X	X	X	√	X
Qin et al. [13]	X	X	X	√	√	X	X	X	X	X	√	X	X	X	X	X	√
Tangpong et al. [14]	X	X	X	√	√	X	X	X	X	X	√	X	X	X	X	√	X

√ denotes exist, **X** denotes does not exist

Table 3.2 Features of RATM compared to related work cont.



Features / Related work	Decision					Expectation				Analytics	Data Management			Monitoring		
	Enable user centric responses	Enable rehabilitation	Enable rewarding	Enable punishment	Evaluate Trust computation methods	Select preferred metrics	Define thresholds for preferred metrics	Define history window size	Enable utilization of new Trust computation methods	Enable utilization of new methods for trust computation evaluation	Enable storage of aggregated data	Enable storage of atomic data	Enable replication in storage	Enable new forms of trust data	Enable collecting data from user (e.g. feedback)	Enable automated monitoring (e.g. collect data from traffic)	Enable collecting new forms of data
PTMS	√	X	X	√	√	√	√	√	√	√	√	√	√	√	√	√	√
Merekoulias et al. [15]	X	√	X	√	X	X	X	X	X	X	√	√	X	X	X	√	X
Choi et al. [16]	X	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√
Conner et al. [17]	X	√	√	√	X	√	X	X	X	X	√	√	√	X	√	X	√
Suryanarayana et al. [18]	X	√	X	√	X	X	√	X	X	X	√	√	√	X	X	X	X
Refaee et al. [19]	X	X	X	√	X	√	X	X	X	X	√	√	√	X	√	√	X
Zaki [20]	X	X	X	√	X	√	X	X	X	X	√	√	√	X	√	X	X
Caverlee [21]	X	X	X	√	X	√	X	X	X	X	√	√	√	X	√	√	X
Khan et al. [22]	X	X	X	√	X	X	X	X	X	X	√	√	√	X	X	√	X
Dai et al. [23]	X	X	X	√	X	X	X	X	X	X	√	√	X	X	√	X	X
Chen et al. [24]	X	X	X	√	X	X	X	X	X	X	√	√	X	X	√	X	X
Xie et al. [25]	X	X	X	√	X	X	X	X	X	X	√	√	X	X	√	X	X
Lopez et al. [26]	√	X	X	√	X	X	X	X	X	X	√	√	X	√	X	√	X
Roy et al. [27]	X	X	X	√	X	X	X	X	X	X	√	√	X	X	√	X	X

√ denotes exist, **X** denotes does not exist

Chapter 4

4. Personalized Trust Management System

In this chapter we present a personalized trust management system, based on RATM. We first illustrate the system model and interaction among its five components. Then we illustrate two trust computations that are used in the proposed system; namely, statistical trust computation and human inspired trust computation. Finally, we discuss the scalability issue in the system and propose our solution to the problem.

4.1. Peer-to-peer Implementation

We utilized the proposed reference architecture to implement a peer-to-peer PTMS. Figure 4.1 illustrates the trust management system using a peer-to-peer model. In the figure, we may view that the five components of the architecture are deployed on each node in the network (either physically or virtually). Each component carries a certain role on a node and communicates and coordinates with its counter on other nodes in the network to form an overlay network for Trust Management. A node in the network may elect to switch off any of its component(s) and use counterpart(s) on remote nodes instead, such as a local server node. Accordingly, the network can be configured to hold a centralized node providing DS, EM, AN and DM services, while at the same time having MM running on each node collecting trust data.

In the system, we implemented DS to receive service selection requests as notification of trustor's requirement for transacting. On receiving such a request, DS informs EM of trustor's requirement, and EM in return operates and replies back with the trust report. The trust report serves as the basis according to which DS makes decisions; however, this does not prevent DS from requesting additional analysis or data using AN and DM in order to make its decisions. DS is responsible for performing all the following actions: service selection, feedback provisioning, applying punishments and rehabilitations. Simply, each decision action is implemented in the form of a method, which DS runs. Here, it is the trustor's responsibility to select the action which he thinks suitable among a set of alternative action methods.

EM receives the trust report formation request from DS and accordingly contacts AN and DM to prepare values for the preferred metrics identified by the trustor in his preferences Document (PDoc). Here, EM

does not directly operate on raw data found in DM; it rather uses a descriptive model of the trust data, which is generated by AN. Such a descriptive model helps reduce the size of the collected data and thus enhances scalability of the system. More details are to be found in the next sections.

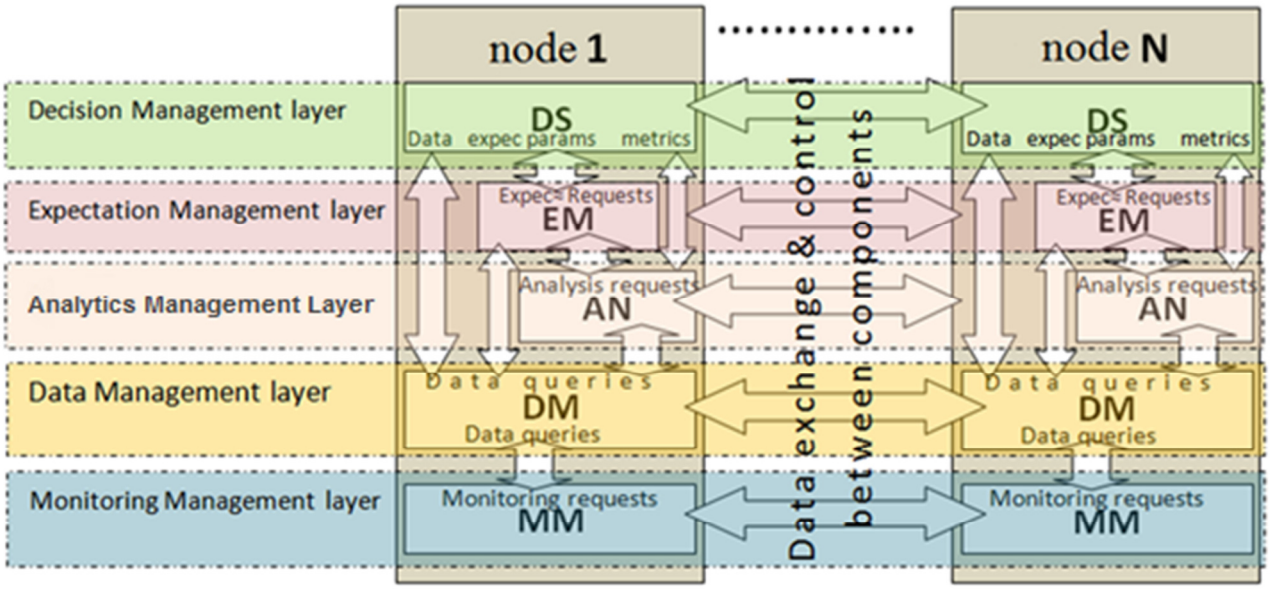


Figure 4.1 Peer-to-peer trust management system overlay network.

AN holds the set of evaluation methods required by EM to calculate the required metrics values from the descriptive model and methods needed to generate such a model. Basically, it operates on the collected data provided by MM that are stored at DM. DM stores the collected data concerning a given trustee in its local Database; each node stores its own data. The descriptive model is stored at a single centralized node in the network and its reliability is accordingly maintained.

MM is responsible for performing data gathering operations. It takes the form of a number of monitoring threads running on each transacting node in the network. These threads may operate on the trustee and/or trustor sides as needed in order to capture necessary data. After transaction termination, monitoring threads forward the collected data to DM for storage.

In the system, we implemented our trust computations in the form of four parameters, analyzing simple network operation parameters. In the next section, we illustrate our trust computation, and then we discuss the scalability issue in our system and present our solution.

4.2. Trust Computation

The proposed trust management system utilizes two main trust computations: statistical trust computation and human inspired trust computation. In the next sections, we illustrate both trust computations.

4.2.1. Statistical Trust Computation

The statistical trust computation mainly depends on calculating statistics, reflecting one or more QoS parameter(s) values, evaluating a service or a party in the network.

Basically, the recorded parameters (by MM) represent data within a time frame window. When a trustor requests selecting a service alternative, the system calculates the Euclidian distance between trustor's preferences and the average of each service alternative's QoS data points (equation 1.1). Data points of all monitored service alternatives are found in the form of a cluster data model. The data point in the model represents sum of scores of QoS parameters recorded from a transaction during the given time window (equation 1.2). Here, the scoring value is computed by dividing the recorded value by maximum possible value for the QoS parameter. The shortest distance service alternative is considered a candidate for transacting (equation 1.3).

The system checks if the shortest distance service alternative QoS parameter values were accepted or not by the trustor.

The following illustrates our statistical trust computation formally:

$$v = eucld(t_i prf, S_{0\cdots n} QoS) \qquad (1.1)$$

eucld(…): function returning Euclidean distance and id of corresponding service

v: 2D vector containing n values for distances and ids

t_i prf: trustor i preferences

$$S_x QoS = \frac{\sum_{i=0}^{n} \sum_{j=0}^{m} S_x RecQoS_j}{n} \qquad (1.2)$$

S_xQoS: service $_x$ average QoS scores

n: total number of transactions

m: total number of QoS parameters

$S_xRecQoS_j$: $service_x$ QoS parameter j score

$S_{0...n}$: $Service_0$ to $Service_n$

$$S_s = id\_of\_\min(v) \qquad (1.3)$$

S_s: candidate service alternative

Id_of_min(…): function calculating minimum value and returning corresponding id

4.2.2.Human Inspired Trust Computation

Inspired by trust in our daily lives [29], we propose using four parameter sets to build trust: Intent, Integrity, Capability and Results. Intent is the cognitive state representing willingness of a party to execute a given action. Integrity is consistency in actions. It is mainly about delivering what is advertised and being consistent in delivering it to different parties. Capability constitutes all resources needed to deliver a service. These resources include hardware, software, skill set and raw material. Results parameters track "promised or contracted" outcomes. Illustration and a computation example for each of the four trust parameters are as follows:

Intent of a party is always issued by a motive. According to that motive, the party places an agenda that includes the actions to be taken in order to achieve that motive. Finally, the party's behavior constitutes execution of these actions. In this scope, Intent of a trustee can be computed using the following:

Trustee's reputation: where having good reputation may constitute the agenda to increase the party's gain.

Trustee's customers' credibility: where having reputable clients may constitute an agenda to have high value contracts and referrals to many new clients.

Warranty: executing warranties may constitute a behavior towards proving the good intent towards delivering a service.

Gain and loss calculations: they are the fundamental motives for transacting and can be measured by comparing popularity and diffusion of different trustees.

The following illustrates an example for calculating the aforementioned parameters:

$$\text{Re}\ p(O) = w * LocOp\ (O) + (1 - w) * ExtOp\ (O, PV) \qquad (2.1)$$

Rep(O): reputation of trustee O is sum of different parties' opinions in a trustee. An opinion constitutes a set of parameters for evaluating a trustee.

LocOp(O): user trust's opinion about trustee O.

w: weight of local opinion.

ExtOp(O,PV): external opinion, representing different parties' opinions about trustee O according to user trust perspective.

$$ExtOp\ (O) = \sum_{i=0}^{i=M} \frac{1}{N} \sum_{p=0}^{p=N} PV\ (p) * fb\ (i, p) \qquad (2.2)$$

M: total number of evaluation parameters chosen by user trust.

N: total number of evaluations provided by different parties concerning trustee O.

PV: vector holding weights, representing user trust's interest in each evaluation parameter.

fb: vector holding evaluation parameters values provided by different parties in trustee O.

Popularity: it is the number of parties who got good service from trustee in the current time sample (figure 4.2), to the total number of parties.

$$Popu\ (Tu,1) = \frac{1}{N} \sum_{i=0}^{i=s} CFun\ (\sum_{j=0}^{j=T} DL_j(SL_i, Tu)) \qquad (2.3)$$

Tu: trustee

Popu(Tu,1): popularity of Tu at time 1

N: total number of parties in the current time sample.

S: number of parties who interacted with trustee during current time sample.

SL: list of parties who interacted with trustee during current time sample.

SLi: party i who interacted with trustee during current time sample.

T: number of transactions between party *i* and trustee in current time sample.

DL: list of delivery status of trustees in transactions {1 delivered, -1 not delivered}

DLj (SLi,Tu) delivery status of trustee *Tu* in transaction *j* with party SL_i in current time sample

CFun: function returns 1 for +ve value and 0 otherwise.

Diffusion: average popularity across history

$$Dif\ (Tu) = \frac{1}{TSN} \sum_{i=1}^{i=TSN} pop\ (Tu, i) \qquad (2.4)$$

Tu: trustee

TSN: the number of time samples = History/TS

TS: time sample length (figure 4.2)

Credibility of customers: sum of diffusion of all parties who were served by trustee.

$$CredT\ (Tu) = \sum_{i=1}^{i=s} Dif\ (SL_i) \qquad (2.5)$$

Tu: trustee

S: the number of parties served by trustee

SL: set of parties served by trustee

Dif(SL$_i$): diffusion of party *SL$_i$*

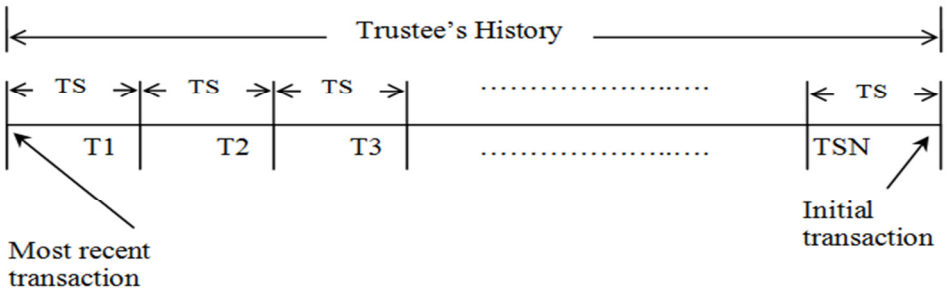


Figure 4.2 Time diagram illustrating trustee's history.

Warranty Quality: it is the reputation of warranty and can be calculated using equation 2.1.

Integrity: of a party is mainly about delivering what the party advertised and being consistent in his/her delivery to all parties he/she interacts with. Consistency implies predictability of party's future behavior. Integrity parameters may include satisfaction of different parties in service delivery, where parties' expectations in a service is to deliver what it advertised. Satisfaction can be measured by collecting opinions of different parties as illustrated in equation 2.1.

Capability: of a party mainly includes all the resources needed to deliver a service. These resources include hardware, software, skill set and raw material. Capability can be measured using 2 metrics; the first identifies the existence of the needed resources at the party under evaluation and the second measures quality of the existing resources. Quality of Resource (QoR) can be measured by reputation of resource and certificates from benchmarking bureaus or organizations, evaluating the resource. Reputation of a resource can be measured using equation 2.1.

Results: of a party is the history of production of that party. Results of a service are what it delivers to its consumers in transactions along its history. Both quality and quantity matter. Thus, results may be measured using the weighed average of metrics of Quality of Product (QoP) and quantity. QoP can be measured by reputation of a product and certificates from benchmarking bureaus or organizations, evaluating the product. Reputation of a product can be measured by using equation 2.1.

4.3. Scalability

Scalability is the ability of a system, a network, or a process to handle a growing amount of work in a capable manner or its ability to be enlarged to accommodate that growth. Personalization, as a concept, maintains individual user's requirements by selecting the best matching service for these requirements. This implies the storage and the processing of huge amounts of data, resulting in scalability issues. Our proposed system includes two basic features upon which it scales up.

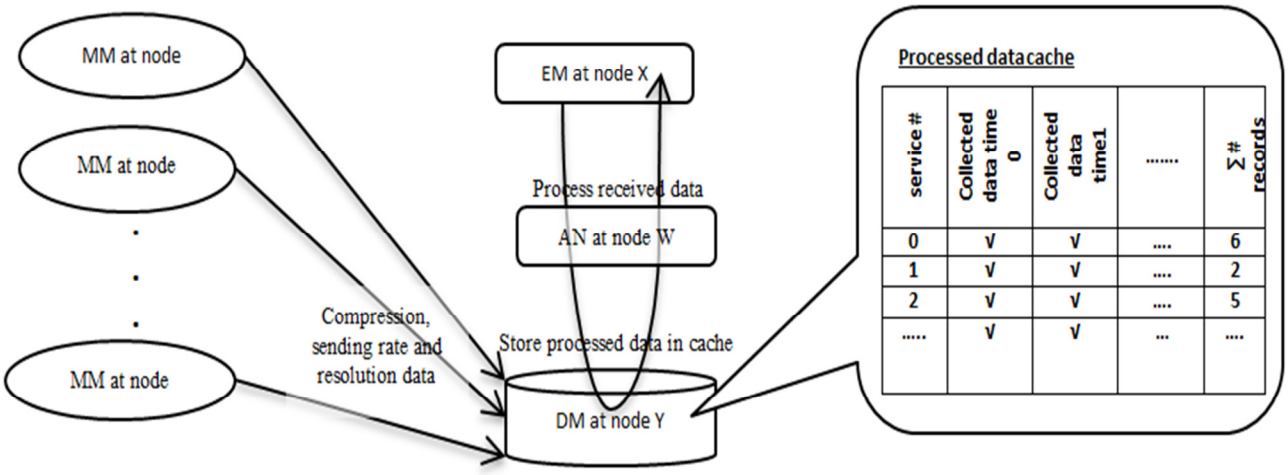


Figure 4.3 Collecting and storing trust data.

The first is the peer-to-peer overlay network structure (figure 4.1), where each peer (node) implements the five main components of a system and each component connects and coordinates with its counter on other peers (nodes) to form an overlay network for managing decisions, expectations, analytics, monitoring and

data. In such a way, additional resources are easily added from the network to the system and thus may easily grow with additional amount of work added.

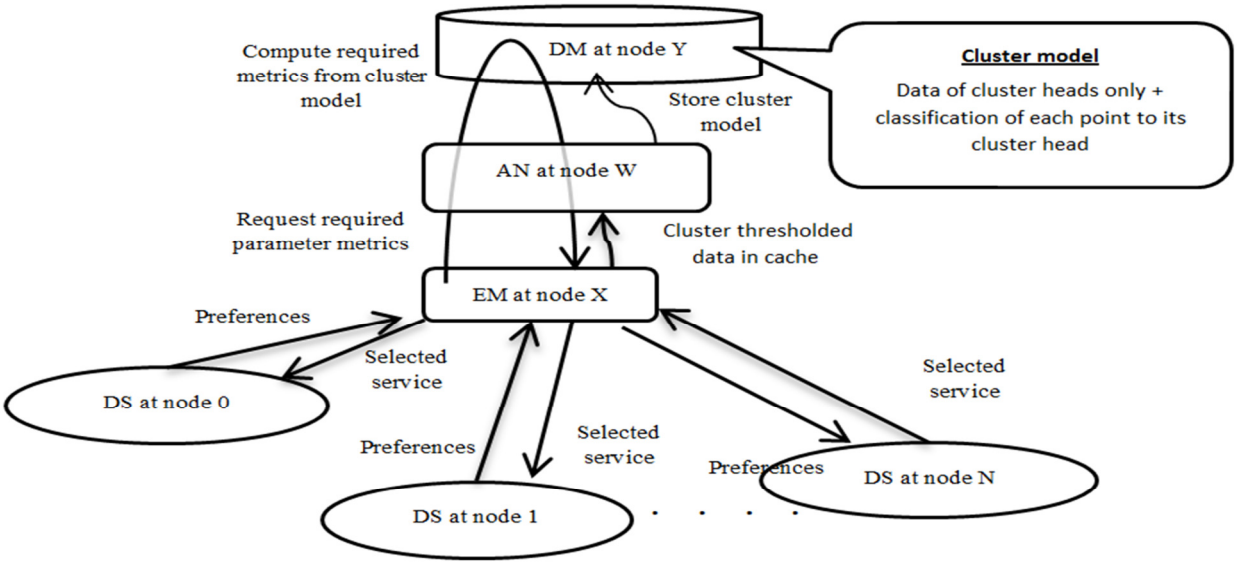


Figure 4.4 Producing trust data clustering model and its utilization in trust decision.

The second feature upon which the system scales is a sampling technique applied to trust data. Such a technique aims at reducing the huge size of data in order to maintain the system's performance. Figure 4.3 and 4.4 illustrates the sampling operation performed by different components of the system. In figure 4.3, different MM components across the network are collecting trust data. The collected data are sent to DM for storage. EM checks for the amount of the collected data in DM, and orders AN to compute/update the total number of records collected about each service and stores it in DM data cache. In figure 4.4, after computing/updating total number of records collected about each service, EM checks if the number of services who accumulated a given number of records passed a certain threshold. If satisfied, EM then orders AN to apply the clustering process to the thresholded data. A clustering model is then produced and saved at DM for service alternative selection operations. The clustering model includes cluster heads data points and classification of non-cluster heads data points. The clustering model here serves as the data model on which analytics, expectation and decision operates. It simply serves as a replacement for the raw trust data that are collected from the network. After producing such a model DM drops all collected data involved in creating the clustering model and only keeps the ones that were not involved for later updates of the model.

4.4. Classification of the proposed PTMS

If we try to classify our proposed trust management system features in the scope of the presented trust classifications in chapter 2, we may find the following:

The proposed system provides an open trust programmability, where the system can be customized and reconfigured to suit user trust requirements.

The system is capable of adopting multiple trust computations and mainly we propose utilizing the human-inspired trust models which constitute a hybrid trust computation of four parameters representing experience-based, credential-based and popularity-based.

The system is open as mentioned earlier and thus may use any form of trust data.

The system utilizes RATM reference architecture and deploys a distributed peer-to-peer model, which mainly benefits scalability of the system.

4.5. Conclusion

In this chapter, we present the proposed peer-to-peer trust management system based on RATM. We further illustrated the two trust computations used, including: statistical trust computation and human inspired trust computation. Additionally, we discuss the scalability issue and presented the solution which included

- adopting peer-to-peer model, where resources can be easily added to the system,
- sampling of trust data using clustering.

Finally, we illustrated the clustering process performed in the system, including the role of each of the five components.

Chapter 5

5. Evaluation

In this chapter we present our evaluations for two case studies: BitTorrent file sharing application and video conferencing application. In BitTorrent case study, we experimented for only high trust preferences diversity. We contrasted effectiveness and efficiency metrics of a simple version of PTMS, which adapts according to preferences of peers in the network, against that of number of simple GTMSs each of fixed thresholds matching peers' preferences in network. In Video conferencing case study, we contrasted effectiveness and efficiency metrics of a more sophisticated version of PTMS against a GTMS version which selects average service alternative in network based on trustors feedbacks and evaluated PTMS's PTMS's scalability. Unlike BitTorrent case study, we experimented for three different trust preferences diversities, namely: low, medium and high. Additionally, for evaluating flexibility of PTMS, we utilized the human inspired trust computation illustrated in section 4.2.2, and the statistical trust computation illustrated in section 4.2.1 in a single PTMS and compared effectiveness and efficiency results for both computations. For the experiments of both case studies, we present: their setup, used trust preferences, role of each component in the trust management systems and finally results and their analysis. Finally at the end of this chapter we provide a discussion illustrating when to choose a personalized trust management system instead of an un-personalized one.

5.1. BitTorrent Case Study

In this section we present our evaluation of trust personalization using BitTorrent file sharing application.

5.1.1.Experimental Setup

To demonstrate our approach we evaluate two versions of a simple trust management system, namely, ***PTMS*** and ***GTMS***. The GTMS defines fixed preferences for all leechers in the network and accordingly provides evaluations in the swarm, while PTMS considers each leechers' preferences and accordingly provides the evaluation separately for each case. Evaluations provided, help leechers to select the suitable leeching service. We used four trust preferences in our experiments illustrated in figure 5.1.

We may describe the leeching service as follows:

Service Type: File sharing -> type: BitTorrent -> type: leeching -> instance: 100Mb file -> [25 providers]

For each one of the 25 instances there exist a service provider and number of service consumers. Leechers perform both roles, service provisioning and service consumption. Service provisioning in the swarm is not homogenous, where we defined 20% of leechers to provide malicious instance of the service. The malicious instance aims at poisoning other leechers' downloaded pieces, where with a certain probability, a bogus chunk is sent to the consumer instead of the original chunk. We used uniform distribution to assign probability of cheating for different malicious instances.

The utilized trust management systems carry the role of optimizing leechers' selection according to the table 5.1.

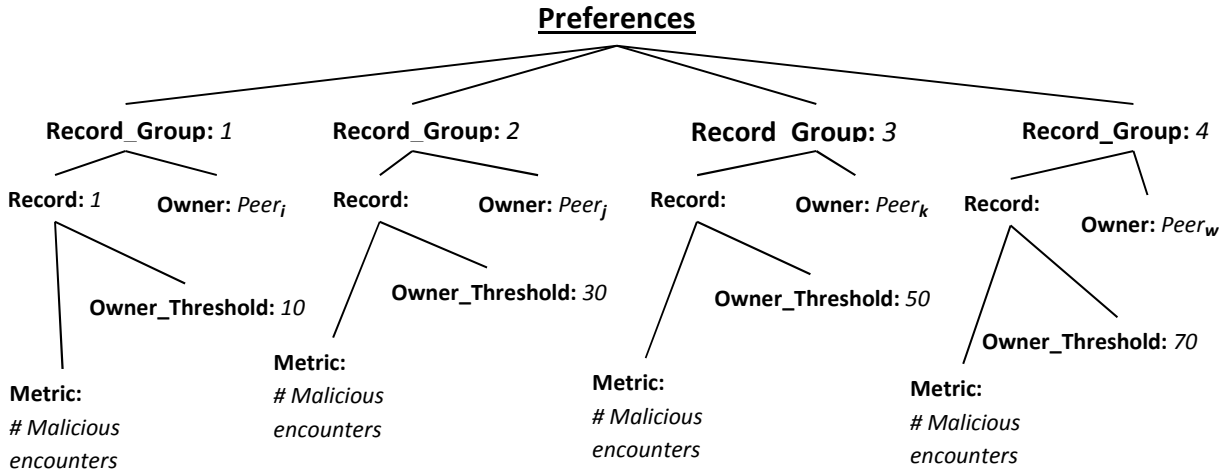


Figure 5.1 Trust preferences for BitTorrent experiments.

Table 5.1 TMS optimization table.

Preference	Accepted Leechers (prob. of cheating)
10	<= 0.2
30	<= 0.4

50	<= 0.6
70	<= 0.8

We may describe trust management operations of both trust systems as follows,

- **MM:** counts number of malicious encounters occurred to the leecher
- **DM:** locally stores each leehcer's preferences and number of malicious encounter
- **AN:** uses table 5.1 to filter peers (compute and identify malicious peers)
- **EM:** in personalized version identifies leecher's owner threshold value and assigns it to AN to operate. In the non personalized version assigns a fixed value through the whole experiment for AN
- **DS:** exit the swarm if number of malicious encounters exceeded TMS threshold (TmsThr)

Each of the five operations is locally performed by each node in the network.

We used collectively six metrics for measuring effectiveness, efficiency and resilience of both versions of trust systems. We used Success rate (Sr) and Download time (Dt) for efficiency and Red-zone exposure duration (Red), Wasted (Wst), Download loss, (Dl), and Overhead (Ovh) for effectiveness. For resilience we re-experimented using 30%, 50% and 70% maliciousness and recoded rate of change in results for maliciousness value.

- ***Red:*** represents percentage amount of time the Leecher spends in the network immediately after his total encounters exceed his Personal trust level (***Ptl***) up till he terminates download. During this time the Leecher is exposed to attack from other Leechers.
- ***Wst:*** represents percentage amount of file downloaded by a Leecher while being in Red. This amount represents wasted effort to the network due to the fact that, the Leecher didn't acquire it, as it should have already departed the network and due to the fact that another Leecher, having larger Ptl value or haven't experienced Red yet, may need this amount.
- ***Dl:*** represents percentage amount of file not downloaded.
- ***Sr:*** represents amount of Leechers who finished downloading the file to number of Leechers who downloaded the file given malicious free swarm.
- ***Dt:*** represent amount of time consumed to download 100% of the file.

- ***Ovh:*** folds of simulation time (10^7) required to distribute the content to the same number of peers who successfully downloaded the same content in the same swarm with no malicious behavior.

We used ns2 simulation environment for implementing our BitTorrent experiments. In each run we calculated Wst, Red, Dl and Sr for Leechers who finished or terminated download. Each run was repeated 5 time using 5 different random seeds and we calculated the average for the six metrics. Table 5.2 illustrates simulation parameters for BitTorrent.

Table 5.2 BitTorrent simulation parameters.

Parameter	Value
# Leechers	30
# Seeds	5
File size	20M
Peers' bandwidth	2048 kbps
# Unchokes	5
#Opt unchokes	1
Peer set size	15

5.1.2.BitTorrent Results

Figure 5.2 to 5.6 illustrates our simulation results for 20% maliciousness. Figure 5.2 represents download loss and wasted for the five runs. In the figure we may observe that TmsThr =70 achieved the least download loss (0%) for all Leechers then TmsThr=10 (5% for Ptl=10, 2% for Ptl=30, 0% for Ptl=50 and 3% for Ptl=70) then TmsThr=30 (21% for Ptl=10, 0% for Ptl=30, 0% for Ptl=50 and 2% for Ptl=70) and finally the personalized trust version (17% for Ptl=10, 5% for Ptl=30, 0% for Ptl=50 and 6% for Ptl=70). Additionally from the same figure we may observe that the least wasted was achieved by TmsThr=10 and the personalized version (0% for all Ptl values) while the highest was for TmsThr=70 (42% for Ptl=10, 7% for Ptl=30, 2% for Ptl=50 and 0% for Ptl=70) then TmsThr=30 (21% for Ptl=10, 0% for Ptl=30 and 50 and 2% for Ptl=70).

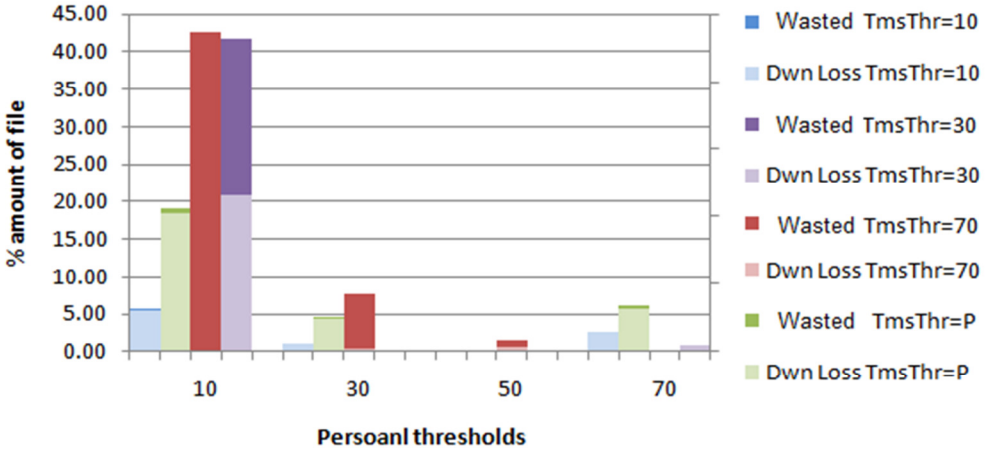


Figure 5.2 Average download loss and average wasted for 20% maliciousness.

Results from this figure favor using the personalized version and TmsThr=10 over other systems. However if we consider figure 5.3 we may observe that TmsThr=70 would be a better choice, as it achieves the highest success rate (60%) compared to that of the personalized trust version (33.97%), TmsThr=30 (33.4%), and TmsThr=10 (25%). If we consider figure 5.4, we may observe that the personalized version achieves the least exposure duration along with TmsThr=10(0% for all Ptl values) while TmsThr=70 achieves the highest exposure duration (31% for Ptl=10, 6% for Ptl=30, 2% for Ptl=50 and 0% for Ptl 70) and TmsThr=30 comes next (9.5% for Ptl=10 and 0% otherwise).

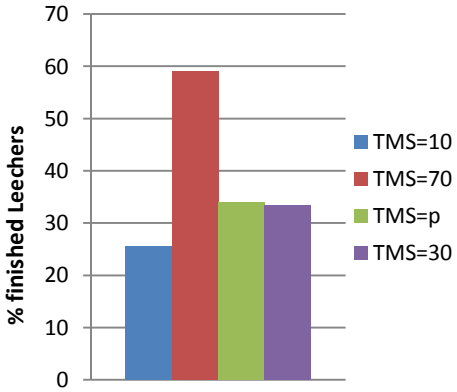


Figure 5.3 Success ratio given 20% maliciousness.

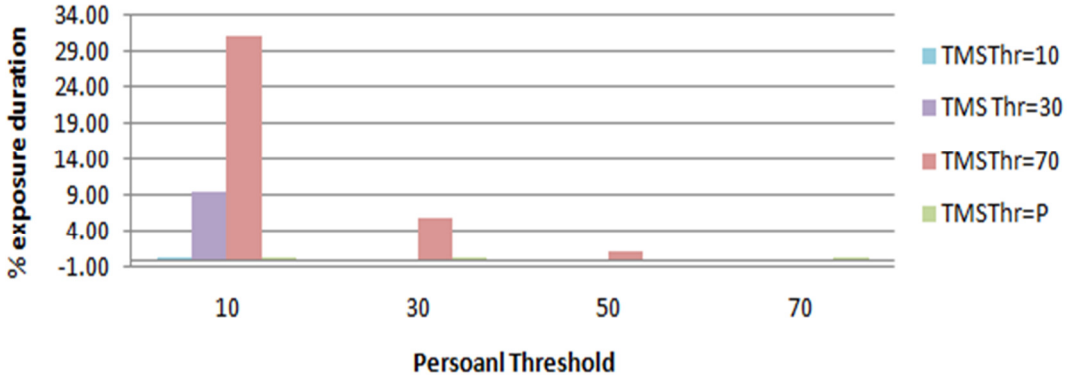


Figure 5.4 Exposure duration for 20% maliciousness.

If we consider figure 5.5 we may observe that TmsThr=30 achieved the least download time (485sec for Ptl=10, 508sec for Ptl=30, 492sec for Ptl=50 and 449sec for Ptl=70) while TmsThr=30 came next (510sec for Ptl=10, 498sec for Ptl=30, 488sec for Ptl=50 and 498sec for Ptl=70) then comes TmsThr=70 (546sec for Ptl=10, 487sec for Ptl=30, 613sec for Ptl=50 and 484sec for Ptl=70) and finally the personalized version was the highest (557sec for Ptl=10 and Ptl=30, 572sec for Ptl=50 and 544sec for Ptl=70). Results from this figure does not favor TmsThr=30 as it still achieves high Dl, Wst and equal Dt to the personalized version.

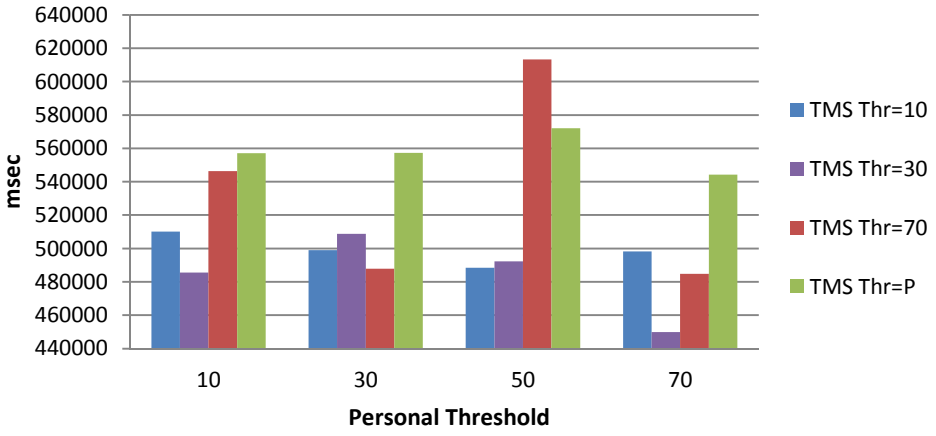


Figure 5.5 Download Time for 20% maliciousness.

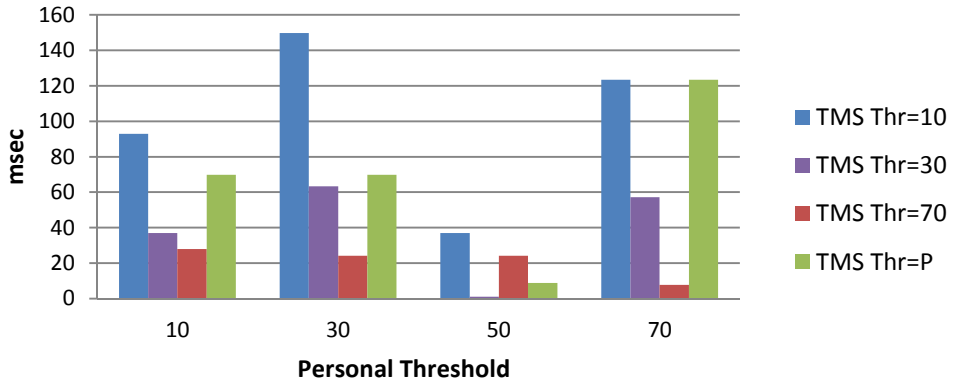


Figure 5.6 Overhead for 20% maliciousness.

Finally from figure 5.6 we may observe that TmsThr=70 achieved the least overhead (28% for Ptl=10, 24% for Ptl=30 and Ptl=50 and 7% for Ptl=70) while TmsThr=30 came next (36% for Ptl=10, 63% for Ptl=30, 0% for Ptl=50 and 57% for Ptl=70) then the personalized system (68% for Ptl=10, 69% for Ptl=30, 8% Ptl=50 and 123% for Ptl=70) and finally TmsThr=10 (92% Ptl=10, 149% for Ptl=30, 36% for Ptl=50 and 123% for Ptl=70). The figure showed little difference between TmsThr=30% and the personalized version leading us to a conclusion that utilizing it in the case of 20% maliciousness in the swarm would be a good compromise which achieves the least Dl, the least Wst, good Sr, and acceptable Ovh.

In order to investigate resiliency of the personalized approach, we repeated the same experiments for 30%, 50% and 70% maliciousness, and calculated average for different Ptl values. Figures 5.7 to 5.12 represent the results.

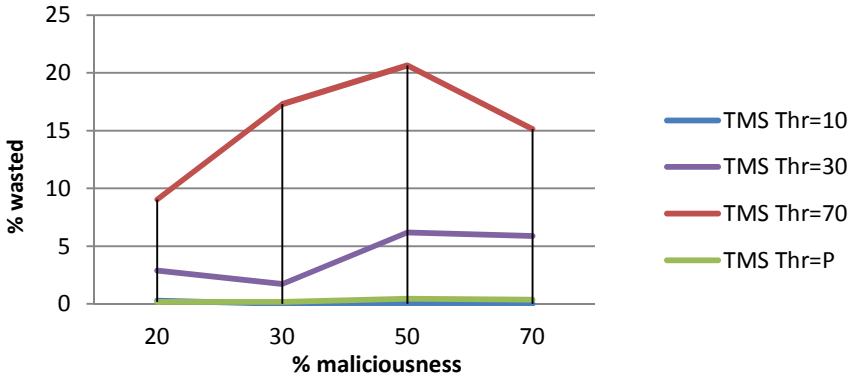


Figure 5.7 Average wasted for 20%, 30%, 50% and 70%.

In figure 5.7, we may observe that the personalized version and TmsThr=10 achieves zero wasted for different maliciousness values, while TmsThr=70 achieved the largest wasted followed by TmsThr=30. The figure shows an increase for TmsThr=70 till 50% maliciousness then a decrease from 50% to 70%. The observed increase is proportional to that of probability of receiving bogus chunk in the swarm, where the high probability cause peers to start their exposure duration earlier than before. Along with the large TmsThr value (70) which maintains a long time period for download before abortion. Such state of increase keeps getting lower and lower with the increase of malicious behavior in the swarm. What prevents the curve from not decreasing is the narrowing of the network allowing fewer peers to make use of existing seeds. The reduction is caused by the quick consumption of the 70 encounters constant causing peers to quickly abort the swarm, giving no time for seeds to make-up for the losses.

In the same figure we may observe a slight decrease for TmsThr=30 from 20 to 30% maliciousness, then it show an increase from 30 to 50 and then stay constant. The decrease from 20 to 30% is caused by the early abortion of peers where they start their red zone early and the short TmsThr constant (30) does not allow them much time to download as it is in the case of TmsThr=70. The increase from 30 to 50% is caused by the thinning of the network allowing fewer peers to connect to existing seeds, which make-up =for the loss caused by early start of red zone. The constant state from 50 to 70% is an equilibrium between the early start of red zone and number of peers connected to seeds.

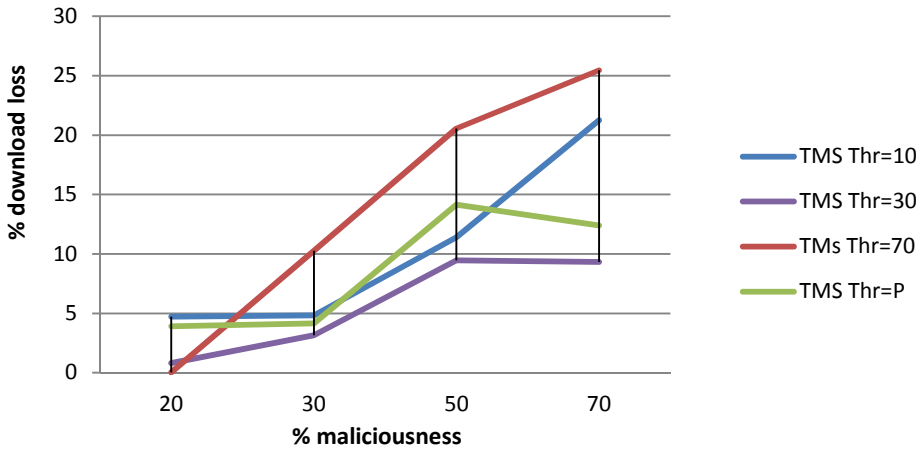


Figure 5.8 Average download loss for 20%, 30%, 50% and 70%.

In figure 5.8 we may observe an increase for all TmsThr values with the exception of the personalized approach which starts decreasing after 50%. Again the increase is proportional to that of probability of receiving bogus chunk, which keeps causing peers in the swarm to start their exposure duration earlier than before. The decrease observed in the personalized system is due to the proper management and utilization of the swarm which isolates malicious peers and saves seeds bandwidth to only well behaving peers.

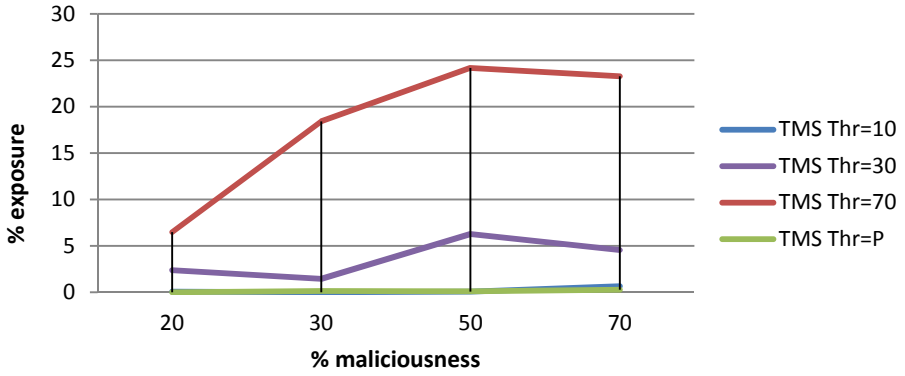


Figure 5.9 Average exposure duration for 20%, 30%, 50% and 70%.

In figure 5.9 we may observe the same increase and decrease as in figure 5.7. In figure 5.10 we may observe an expected decrease in all cases. In figure 5.11 we may observe an expected increase for all cases except the personalized system which decreases from 50 to 70. Again the decrease is due to the proper management and utilization of the swarm which isolates malicious peers and saves seeds bandwidth to only well behaving peers.

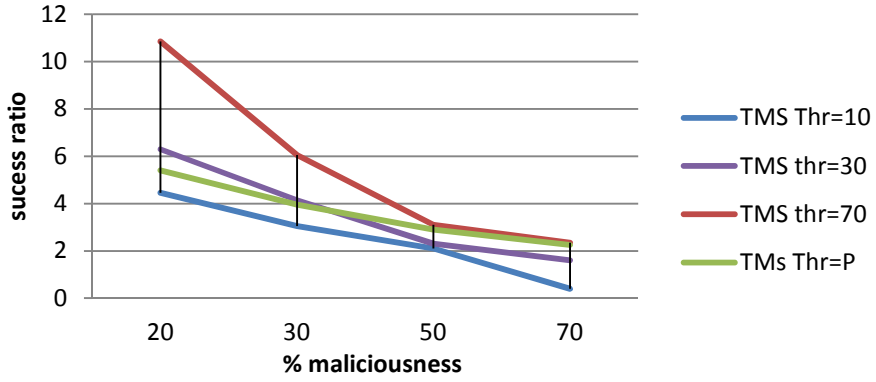


Figure 5.10 Average success ratio for 20%, 30%, 50% and 70%.

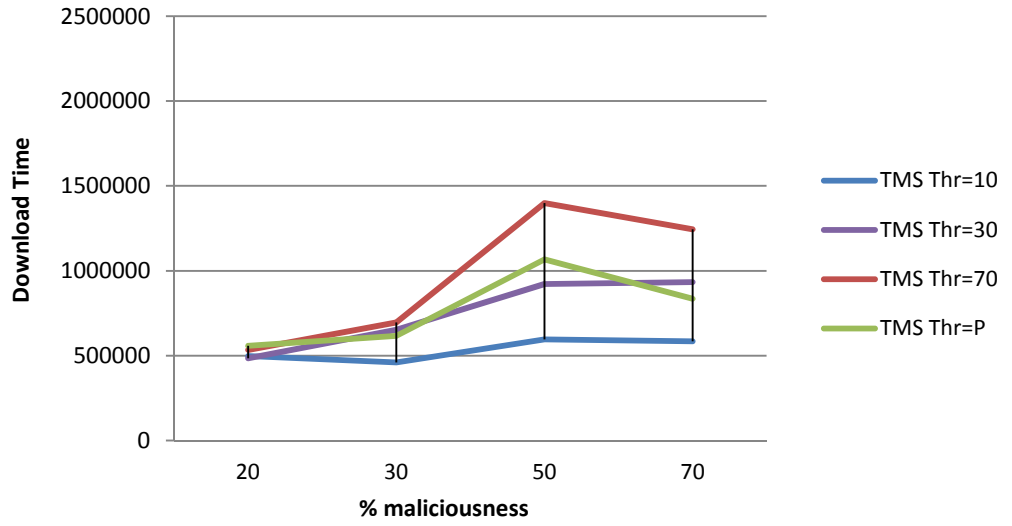


Figure 5.11 Average download time for 20%, 30%, 50% and 70%.

In figure 5.12 we may observe slight overhead increase except for TmsThr=10 which depicts a dramatic value increase from 50 to 70%. This is expected as the success ratio recodes the lowest value for TmsThr=10.

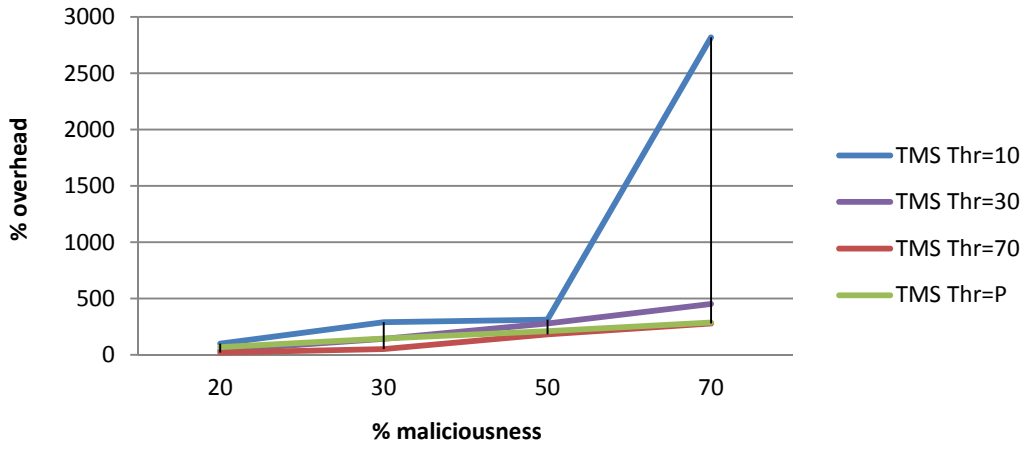


Figure 5.12 Average overhead for 20%, 30%, 50% and 70%.

If we examine figures 5.7 to 5.12 and find the differences between results values at each maliciousness point, we may find that the personalized system achieves the least difference among all systems in case of wasted, exposure, success ratio, overhead and achieves equal difference values with TmsThr=30, in case of download loss. Such observation depicts resilience of the personalized system.

5.1.3.Findings

From the observations and aforementioned discussion, we may conclude that,

- The utilization of PTMS enhanced leechers' QoS level of satisfaction, where PTMS produced the results which, when compared to that of GTMS, constituted a good compromise in terms of overhead, exposure duration, wasted and success ration, for all leechers given their diverse trust thresholds.
- Trust personalization increases resiliency to attacks, where with increased amount of malicious behavior in the network, effectiveness and efficiency results for PTMS outperformed that of different GTMSs for various degrees of maliciousness.

5.2. Video Conferencing Case Study

5.2.1.Overview of Case Study

In this section we present our evaluations of a more sophisticated RATM-based system. Our evaluation approach is based on measuring effectiveness and efficiency of a PTMS operating in a simulated heterogeneous marketplace of video streaming services. In the marketplace, we stochastically created parameterized trust-driven transactions for satisfying video streaming senders' trust requirements. Our simulation model included a centralized server providing trust management services and video stream sender and receiver nodes. Sender nodes use trust management services to select the most suitable video streaming service among 64 available service alternatives. For each service alterative, we provided a set of three assets according to which it operates and delivers the metrics values in table 5.3. These assets are,

- the compression procedure, according to which it generates the Compression (C) value,
- the congestion procedure, which includes choosing the Sending Rate (SR) value
- and finally the video creation procedure, which delivers the Video Resolution (R) value.

We defined service alternatives' behavior as to operate its owned assets, and thus a service alternative delivers according to its capabilities. Video senders' trust requirements were based on variations in values of the three metrics, SR, C and R. SR refer to the number of times a service can reduce its stream rate

after sensing congestion in the network. The higher the value for SR, the better a service can manage with congestions and vice versa. C refers to compression rate of the video. In our simulation, we reduce size of packets sent according to C value. R refers to the resolution value in pixels for the videos. Table 5.3 illustrates different values for the trust requirements metrics.

Table 5.3 Trust Requirements Metrics values

Trust Metric	Metric values
SR	2, 3, 4, 5
C	1%, 3%, 5%, 7%
R	240p, 360p, 480p, 720p

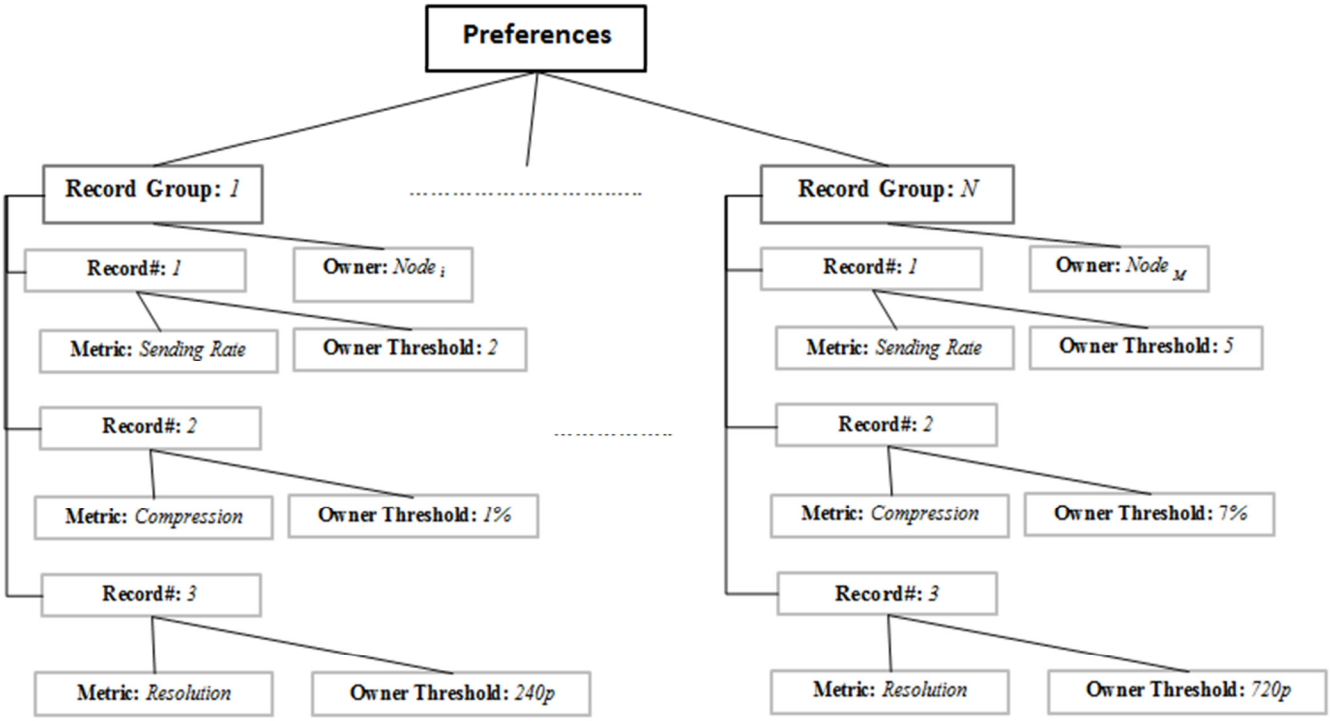


Figure 5.13 Sample trust preferences used in our simulation.

Video streaming services were built to satisfy video streaming senders' trust requirements. We used a total of 64 service alternatives providing all possible combinations of values, in the table, for the three metrics.

Matching between video sender's requirements and the suitable service alternative was achieved by the trust management service. Figure 5.13 illustrates a sample of trust preferences used in our simulation.

In addition to PTMS, we implemented GTMS as a reputation system in which nodes provide their evaluations of service alternatives in the form of feedback. A node provides feedback value 1 in case it received SR, C and R values satisfying its preferences, while provides the value 0 otherwise. The reputation of each service alternative is computed as the sum of feedbacks provided for each service. Here, GTMS is completely oblivious to trustor requirements, where feedbacks from different users — regardless of their diverse trust requirements— are summed to produce a single reputation score value.

5.2.2.Experiments Setup

Our evaluation comprises three sets of experiments. In the first set we measured and demonstrated effectiveness and efficiency of PTMS. While in the second set we verified and demonstrated the impact of personalization in trust management. In the third set we verified and demonstrated flexibility of PTMS.

Based on these sets of experiments we provide two main contrasts. The first demonstrates effectiveness, efficiency and resilience of PTMS, while the second demonstrates PTMS's flexibility including an evaluation for the utilization of the trust parameters, Intent, Integrity and Results. In the first contrast, we compared between effectiveness, and efficiency metrics results of PTMS utilizing; sending rate, compression and resolution metrics, against that of GTMS utilizing the same metrics. In the second contrast, we compared between effectiveness and efficiency metrics results for two different trust computations implemented on the same PTMS. Our simulation was achieved using ns2 [30] simulation environment.

5.2.3.Possible Misbehavior

There are two types of misbehaving nodes, namely, selfish and malicious nodes. A selfish node is aiming at unfairly gaining self-benefit by its misbehavior. This includes all forms of violation to the contract between transaction parties. For instance, cheating and not delivering or providing low QoS parameters. A malicious node is aiming at negatively impacting the system. For instance hijacking an MM component and use it to provide false trust data.

In our simulation we only included selfish misbehavior. We defined the selfish misbehaving service alternative behavior as not to operate its assets and thus do not deliver. For instance, if a service alternative has the assets to deliver 2, 5% and 360p for SR, C and R respectively. The selfish service would deliver 0, 0% and 140p respectively, while the well behaving would deliver 2, 5% and 360p respectively.

We included in simulation an amount of selfish misbehavior, where, we experimented using 10% of service alternatives exhibiting selfish misbehavior, and then re-experimented using 20%. A selfish service alternative would misbehave with uniform probability of 25% during simulation. i.e. selfish services does not misbehave all the time during simulation.

5.2.4.Model and Operation of PTMS

In the case of ***PTMS***, the centralized server implements DS, EM, AN, and DM components of RATM. While an MM component runs on each video stream sender and receiver nodes. In operation, Video stream senders send their trust preferences to the centralized server via request message, and in turn, the server performs the operations, expectation, analytics, data management and decision, and replies with the selected service alternative ID. Each component operates as follow:

MM component:

While being in a transaction, ***MM***, records the required trust data and at the end of the transaction, sends the recorded data to ***DM*** which in turn informs ***EM*** at the centralized server.

EM component:

EM then checks if collected data, in the buffer, at **DM**, had reached a given threshold value, termed Maturity (MAT). MAT refers to the number of collected data points about a set of service alternatives. We computed MAT values by dividing all accumulated history in system's cache, by History Window (HWin) size. In our simulations, we utilized different MAT values according to which we produce our clustering data model. Simply on receiving new data, we compute MAT value and compare it against the desired threshold value, if the threshold is satisfied then we start the clustering data in cache. We used HWin of size 3, and used 24, 34, 54, and 64 threshold values for MAT.

AN component:

AN requests the thresholded data from ***DM*** and are then removed from data buffer. Then ***AN*** computes the data and similarity matrices and passes them to the clustering procedure. We used Affinity

propagation (AP) [31] clustering in our simulations. The clustering procedure's output is then passed to ***DM***.

DM component:

DM stores the clustering model (i.e. to which exemplar, each data point was classified) is then stored at DM along with trust data of exemplars' points only. Here, it is worth noting that the utilization of the clustering data model, instead of raw trust data, greatly helped in reducing the size and complexity of the stored data (results illustrated in figure 5.17).

5.2.5.Model and Operation of GTMS

In the case of ***GTMS***, the simulation included heterogeneous nodes (different in their trust requirements), each of which provides the feedback which reflects their own satisfaction in a video stream service alternative which he/she used. The feedback is collected by a centralized server which provides trust management services. The centralized trust service sums the collected feedbacks, and computes a reputation score for each service alternatives in the network.

5.2.6.Simulation Scenario

PTMS simulation scenario starts by scheduling, randomly selected sender and receiver nodes, for running a number of video stream transactions. We simulated 100 nodes performing 1000 video streaming transactions. Each transaction starts by the sender node checking if it has a selected satisfying service alternative for video streaming.

DS component:

If not, the sender node sends a message holding its preferences, to the ***DS*** of the centralized server, requesting selection of video stream service alternative. In turn, the ***DS*** requests ***EM*** to develop trust report about service alternatives.

EM component:

To achieve that ***EM*** checks for stored data in ***DM***. If no data were found (in case of boot strapping), ***EM*** informs ***DS*** of bootstrapping status and selects random service alternative. Otherwise, in case of available data at ***DM***, ***EM*** calculates, using ***AN***, the Euclidian distance between the received preferences and the average of each service alternative's data points values. ***EM*** then passes the results to ***DS*** for selection. ***DS*** checks if the shortest distance service alternative was accepted or not and sends to user the selected service index in case of acceptance. In case of no acceptance, random service is selected and its index is sent. The service alternative is accepted if the shortest distance service alternative trust profile has the

exact preferences values as the sender node. On receiving ***DS*** reply, the video streaming node starts sending stream packets to its receiver node using the selected service.

MM component:

During send/receive operations, ***MM***, on both sides, records packet sending rate, video packets sizes, and resolution data. On stream termination, ***MM*** sends recorded data to ***DM*** which informs ***EM*** at the centralized server where it is prepared for storage and analysis for cluster model update. These operations take place for all the scheduled transaction. When all transactions are done, the simulator logs evaluation data.

GTMS scenario operates much like PTMS scenario. Nodes contact the centralized server to choose a suitable service alternative. After transaction ends, video stream sender nodes provide their feedback to the centralized server. The centralized server sums feedbacks, providing each service alternative a reputation score. Nodes in the network choses the most reputable service to transact with.

At the beginning of simulation the trust server bootstraps by selecting random service alternative for nodes. Feedback during bootstrap is collected. Bootstrap duration lasts utile a given number of feedback messages is achieved, termed bootstrap window. After the bootstrap duration, the trust server operates and provides the reputation scores. We chose 24, 34, 44, 54 and 64 as values for bootstrap window.

5.2.7.Space and Time Complexities of PTMS

Now let us compute space and time complexities for PTMS. As mentioned earlier, PTMS performs 2 basic operations, service alternative selection operation and data storage operations. The two operations are performed concurrently and independent of each other.

Selection Operation

First we start with service alternative selection operation. In this operation, DS component receives user trust preferences and passes them to EM component for trust report computation. EM component in turn requests, from AN computing average of all service alternatives data points found in data model stored in DM. If we assume *NDP* data points, and *M* number of metrics, then we have

$$M*NDP \qquad (5.1)$$

Next EM requests, using AN, measuring distance between each average value and user trust preferences. If we assume we have *NSA* service alternatives, then we have *NSA* steps.

In total we have

$$M*NDP+NSA \qquad (5.2)$$

Finally EM requests, from AN, finding minimum distance value resulting in additional *NSA* steps.

Accordingly, we may find that the selection process **time complexity** is: O(*M*NDP*+2*NSA*)

Now if we consider space complexity, we may find that we have *M*NDP* locations for computing average of each data point, *NSA* for computing distance and an additional location in memory which holds minimum value.

Accordingly we may find that the selection process **space complexity** is: O(*M*NDP*+*NSA*)

Storage Operation:

Next we compute space and time complexities for storage process. MM component is distributed on all nodes in the network, monitoring all transactions. Having M number of metrics we get M locations.

When a transaction is finished, MM sends the collected data to DM for storage. If we assume having *NT* transactions and *X* parties, then we get amount of storage of

$$X*NT \qquad (5.3)$$

In total we get

$$M+X*NT \qquad (5.4)$$

When DM receives the new data from MM, it informs EM, accordingly EM requests, from AN, counting number of services alternatives who achieved a minimum of Hwin collected records. Then the computed result is compared against MAT threshold. If threshold is satisfied then, the thresholded data is then passed to the clustering procedure at AN. If we assume a total of *NDC* data points in DM cache, then we get in this NDC steps. In total we get:

$$M+X*NT+NDC \qquad (5.5)$$

The amount of data passed to the clustering process is *Hwin * MAT*. The clustering procedure performing affinity propagation is

$$(Hwin*MAT)^3 \qquad (5.6)$$

After clustering, DM stores data points for cluster heads and classification of non cluster heads data points.

Accordingly, we may calculate storage process **time complexity** as: O($M+X*NT+NDC+(Hwin*MAT)^3$).

Now if we consider space complexity, we may find that we have number of locations for monitored data

$$M+X*NT \qquad (5.7)$$

In addition to *NDC* for stored data in cache, and for similarity metrics passed for clustering procedure we have

$$Hwin*MAT \qquad (5.8)$$

and CM for clustering data model. Accordingly we may compute selection process **space complexity** as: $O(M+X*NT+NDC+(Hwin*MAT)^2 +Hwin*MAT+ CM)$

5.2.7.1. Discussion on Parallelization of PTMS

The computed space and time complexities for PTMS shows considerably high overhead for the two performed operations, especially the storage operation. Accordingly, we may reduce such overhead by parallelizing different components in PTMS. So if we consider the selection operation first, we may view that the following operations may be reduced when distributed across N number of nodes:

- Average of each service alterative data points, we may get

$$M*NDP/N \quad (5.9)$$

- Distance measure between user trust preferences and each service alternative data point, we may get

$$NSA/N \quad (5.10)$$

- Finding minimum value for distance measure, we may get

$$NSA/N \quad (5.11)$$

Accordingly we may have **time complexity** reduced by N having: $O((M*NDP+2NSA)/N)$

For **space complexity**, if we adopt N storage locations we may reduce space complexity by N getting space of $O((M*NDP+NSA)/N)$

Now if we consider storage operation which incurs more overhead, mainly due to the clustering operation. We may find that, if we implement a parallel version of affinity Propagation by distributing data points across N nodes. We may easily implement the message passing algorithm necessary for exchanging similarity, responsibility and availability of data points, thus reducing time complexity into

$$(Hwin*MAT/N)^3 \quad (5.12)$$

Hence the overall **time complexity** can be calculated as $O((M+X*NT+NDC)/N+(Hwin*MAT/N)^3)$

The main disadvantage of such approach is that we are increasing number of message necessary for distributing and retrieving data across the network resulting in higher traffic across network.

If we consider **space complexity**, again we may distribute required space across the network resulting in N reduced space. Accordingly we have $O((M+X*NT+NDC+Hwin*MAT+CM)/N+(Hwin*MAT/N)^2)$

5.2.8.Space and Time Complexities of GTMS

Now let us compute space and time complexities for GTMS. Again we have two operations selection and storage.

Selection Operation:

Basically the selection operation depends on selecting the max score service alternative. This includes trust score for each service alternative. Therefore, if we have NSA service alternatives then we may calculate **time complexity** for selection operation of GTMS as O(NSA).

Space complexity includes storage location for each service alternative score. Accordingly we may calculate **space complexity** for selection operation of GTMS as O(NSA).

Storage Operation:

The storage operation collects data evaluating each transaction from transaction parties. If we assume *X* transaction parties and *NT* number of transactions, accordingly we may calculate **time complexity** for storage operation as $X*NT$ (5.13)

For space complexity, we have *NSA* scores in addition to

$$X*NT \quad (5.14)$$

data for evaluating transactions. Accordingly we may calculate **space complexity** of storage operation as: O($NSA+X*NT$).

5.2.9.Evaluation Metrics

We evaluated **PTMS** service using a set of metrics measuring effectiveness and efficiency. An efficiency metric measures how good the trust management service is. While an effectiveness, metric measures how useful the system is as a trust management service.

To measure efficiency, we recorded,

- Number of bootstrap messages,
- Message-transaction ratio
- and amount of used data by the system.

We calculated number of bootstrap messages by, counting service-selection-request messages received by the centralized server, from the start of simulation and up till the very first non-random selection of the server. We calculated message-transaction ratio by counting number of received messages by the centralized server during simulation and dividing it by number of conferences (1000 conferences). We did the same for sent messages from server to sender nodes. We calculated amount of used data by the system by the ratio between the number of data exemplar points and the total number of collected data points.

To measure effectiveness, we recorded amount of service alternative selection messages which caused satisfaction state to total number of selection messages, termed selection usefulness.

For the **GTMS** system, we measured message-transaction ratio and selection usefulness.

5.2.10. Results

In this section we illustrate our results which comprises of two main contrasts. The first contrast demonstrates effectiveness, efficiency and resilience of PTMS. While the second demonstrates PTMS's flexibility including an evaluation for the utilization of the trust parameters, Intent, Integrity and Results. In the first contrast, we compared between effectiveness, and efficiency metrics results of PTMS utilizing; sending rate, compression and resolution metrics, against that of GTMS utilizing the same metrics. In the second contrast, we compared between effectiveness and efficiency metrics results for two different trust computations implemented on the same PTMS. Trust computation here is performed by combined operations of RATM components, DS, EM and AN. The first trust computation utilizes, Sending Rate, Compression and Resolution metrics for trust computations, while and the second utilizes, Intent, Integrity and Results metrics. The following sections illustrate our work.

5.2.10.1. PTMS vs GTMS

Figure 5.14 to 5.20 illustrate effectiveness and efficiency metrics results for PTMS vs GTMS. From the figures we may observe that, high effectiveness and efficiency values are encountered at high MAT values given 10% and 20% selfish misbehavior in the network. In addition, the very small difference in results values between 10% and 20% selfish misbehavior, illustrate PTMS resilience against the misbehavior.

Figure 5.14 illustrates average number of messages needed by a system to start providing reliable selections, given 10% and 20% selfish misbehavior. In the figure, for PTMS 10% selfish misbehavior, number of messages for bootstrapping starts at 73 messages received by the server at 24 MAT and gradually increases till reaches 89 message at 64 MAT. The 20% selfish misbehavior shows the same results. Here at bootstrapping state, the server checks for accumulated data, if MAT threshold value is achieved, all data satisfying MAT is passed to the system and selection operation starts. If not achieved, a random service is selected. Such observed increase is normal where the amount of collected data from the network increases when increasing history window size.

Figure 5.15 illustrates message-transaction ratio for received messages by the centralized server. The figure shows results for 10% and 20% selfish misbehavior for PTMS and GTMS. In the figure, for PTMS

and 10% selfish misbehavior, the received message count starts at 2.1 and gradually decreases while increasing MAT values till reaches 1.3. While in the case of 20% selfish misbehavior message count shows almost the same values. The decrease in message count while increasing MAT values shows an increase in number of satisfied nodes, hence the reduced number of service alternative selection messages sent to the server.

In case of GTMS, we may observe lower values than that of PTMS at low MAT values. Where, if we consider 10% selfish misbehavior, we may observe that the received message count starts at 1.75 for 24 MAT. Then decreases to 1.5 at 34 and 44 MAT. Then increases to 1.55 at 54 MAT and finally reaches 1.6. Here as we can see, the lowest value was recorded at 44 MAT, while the values are higher after and before this point. The higher values observed after are due to the less satisfaction in the network. This is caused by the increase in number of heterogeneous feedbacks, which causes more distortion in the produced reputations values. While the high values observed before is also due to the less satisfaction in the network. This is caused by the unreliable amount of data (feedbacks) collected from the network.

In the case of GTMS 20% selfish misbehavior, the received message count starts at 1.6 and gradually decreases till reaches 1.5 for 34 and 44 MAT, then decreases at 54 MAT to reach 1.3 and finally increases to reaches 1.5.

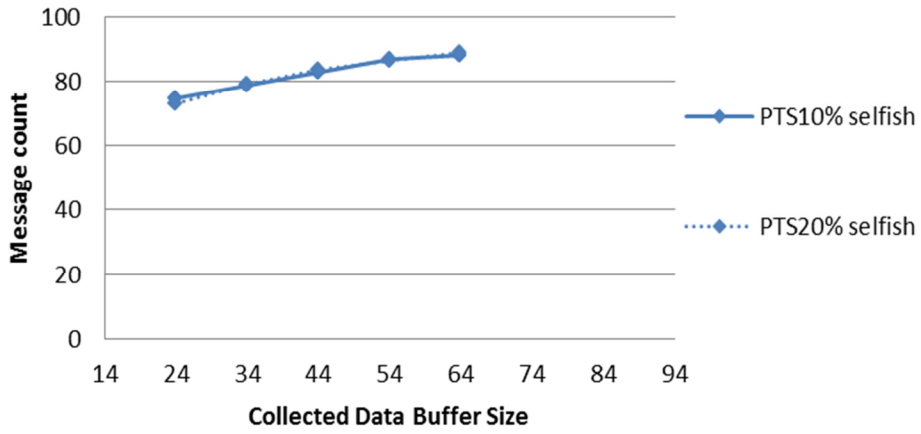


Figure 5.14 Bootstrapping messages for different MAT values.

If we compare the 10% and 20% selfish misbehavior results for GTMS, we may observe that message count are almost the same for 24, 34 and 44 MATs, while differs at 54 and 64 MAT. At 54 MAT the 10% results produces higher number of messages than that of the 20%. This difference shows that in the 10%

case, satisfaction state is reached at a later point of time than that of the 20% case. This indicates that the bootstrapping window size 44 is the optimal case (achieved least number of messages) for 10% misbehavior. Here, increasing or decreasing bootstrapping window size would cause more messages to exist in the network. While in case of 20% misbehavior, the optimal window size is 54 at which the least amount of messages is achieved.

If we compare PTMS and GTMS results we may find that only in the case of 64 MAT, PTMS produces lower number of messages than that of GTMS. The low number of messages sent to server in simulation is an indication of amount of service alternative selection messages sent by nodes. This is due to the fact that, a node when not achieving a satisfying state, sends/resends service alternative selection message. Therefore, the results in figure 5.15 interprets that PTMS, at 24, 34 and 44 MAT, achieved nodes' satisfaction state at a point in simulation time later than GTMS, except for the case of 64 MAT. At 64 MAT PTMS achieved nodes satisfaction before GTMS could.

The difference between PTMS and GTMS at 54 and 64 MAT values is counter intuitive, where it was expected, that message count would record higher values in the case of PTMS than that in the case of GTMS due to the overhead caused by personalization. However, the contrary was observed. This is due to the fact that video senders, in the case of PTMS, received satisfying services alternatives, resulting in fewer amounts of service selection requests being sent to the server. While in the case of GTMS, selection usefulness is low, resulting in high number of request messages being sent to the server.

The results for PTMS showed almost identical values for both cases 10% and 20%, depicting resilience against the selfish misbehavior.

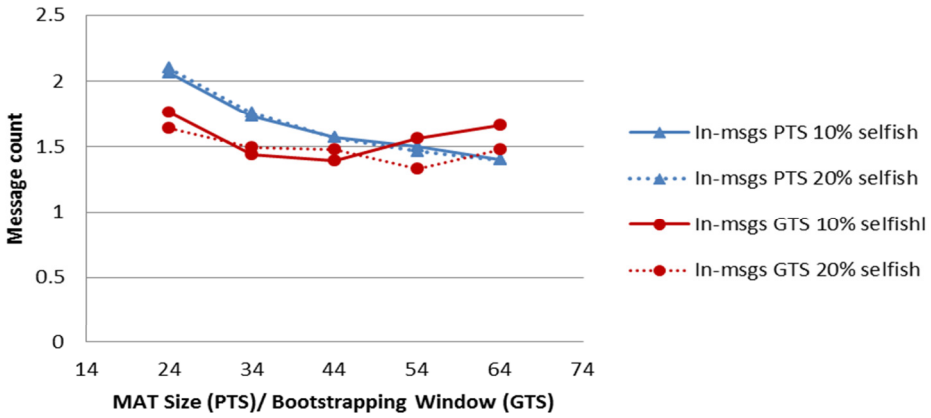


Figure 5.15 Message-transaction ratio for received messages (In-msgs), for different MAT values.

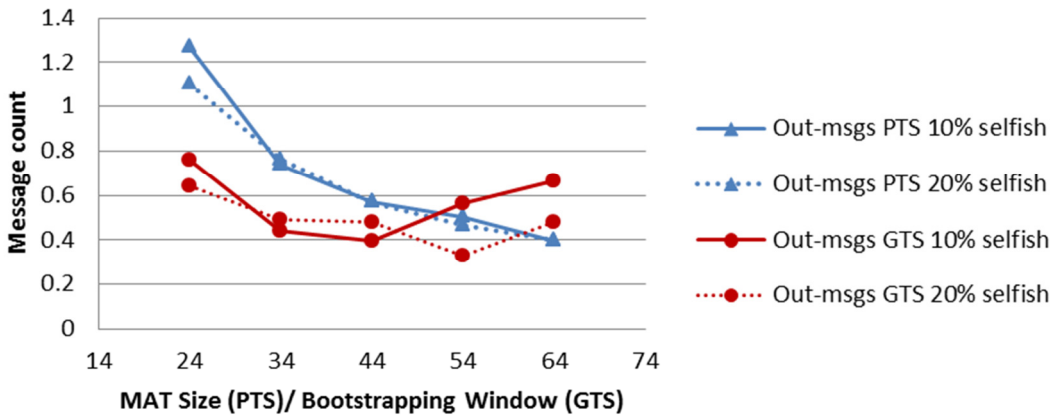


Figure 5.16 Message-transaction ratio for sent messages (Out-msgs), for different MAT values.

Figure 5.16 illustrates message-transaction ratio for messages sent by the centralized server. The figure shows results for 10% and 20% selfish misbehavior for PTMS and GTMS. In the figure, for PTMS 10% selfish misbehavior, average number of messages starts at 1.27 at 24 MAT and then decreases to 0.75, then 0.57, then 0.5 and finally reaches 0.4 at 64 MAT. PTMS 20% selfish misbehavior shows almost identical results except for the start at 1.1. The decrease in message count while increasing MAT values shows an increase in number of satisfied nodes, hence the reduced number of service alternative selection messages sent to the server.

In case of **GTMS**, we may observe lower values than that of PTMS at low MAT values. Where, if we consider 10% selfish misbehavior, we may observe that the received message count starts at 0.78 for 24 MAT. Then decreases to 0.45 at 34 and 0.4 at 44 MAT. Then increases to 0.58 at 54 MAT and finally reaches 0.7. Here as we can see, the lowest value was recorded at 44 MAT, while the values are higher after and before this point. The higher values observed after are due to the less satisfaction in the network. This is caused by the increase in number of heterogeneous feedbacks, which causes more distortion in the produced reputations values. While the high values observed before is also due to the less satisfaction in the network. However this is caused by the unreliable amount of data (feedbacks) collected.

In the case of GTMS 20% selfish misbehavior, the received message count starts at 0.65 and gradually decreases till reaches 0.5 for 34 and 44 MAT, then decreases at 54 MAT to reach 0.35 and finally increases to reaches 0.48.

If we compare the 10% and 20% selfish misbehavior results for GTMS, we may observe that message count are almost the same for 24, 34 and 44 MATs, while differs at 54 and 64 MAT. At 54 MAT the 10%

results produces higher number of messages than that of the 20%. This difference shows that in the 10% case, satisfaction state is reached at a later point of time than that of the 20% case. This indicates that the bootstrapping window size 44 is the optimal case (achieved least number of messages) for 10% misbehavior. Here, increasing or decreasing bootstrapping window size would cause more messages to exist in the network. While in case of 20% misbehavior, the optimal window size is 54 at which the least amount of messages is achieved.

If we compare PTMS and GTMS results we may find that only in the case of 64 MAT, PTMS produces lower number of messages than GTMS. Again this is due to the high satisfied number of nodes in the network resending service alternative select messages to PTMS server.

Again the results for PTMS shows close values for both cases 10% and 20%, depicting resilience against the selfish misbehavior.

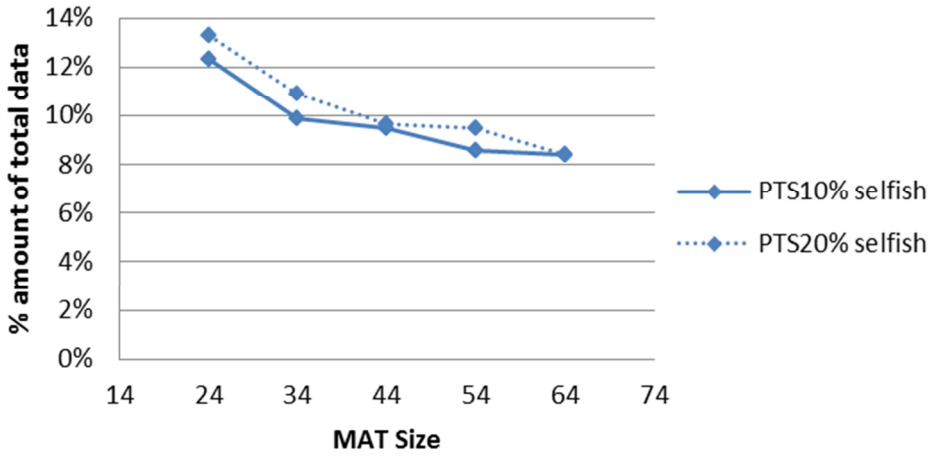


Figure 5.17 Amount of used data by the system for different MAT values.

Figure 5.17 illustrates amount of used data by PTMS to the total collected data for 10% and 20% selfish misbehavior. In the figure, at10% selfish misbehavior, the amount of used data starts at 12.5% at 24 MAT and decreases to 10% at 34 MAT and 9.5% at 44 MAT, and then it reaches 8.5% at 54 and 64 MAT. Here, as we expected the amount of used data by the system decreases gradually when increasing MAT values. This is due to increasing the size of data passed to the clustering process, resulting in more common patterns found by the clustering engine.

In case of 20% selfish misbehavior, the amount of used data starts at 13.5% at 24 MAT and gradually decreases to 11% at 34 MAT, then decreases to reach 9.5% at 44 MAT, then stays constant at 54 MAT and finally reaches 8.5% at 64 MAT. Again as we expected, amount of used data decreases as we increase MAT values.

If we compare the amount of used data for 10% and 20% misbehavior, we may observe that both curves are decreasing gradually with slight constant difference of the value 1% till 44 MAT. At this point both curves have the same value then the constant difference returns at 54 MAT and then the two curves reunite again at 64 MAT. Such constant differences in values are caused by the difference in amount of selfish misbehavior resulting in more data patterns in case of 20% selfish misbehavior. This increase in data patterns are caused by the increase in misbehaving services count, resulting in more data points representing misbehaving services in the cluster model.

The low constant difference value 1% between 10% and 20% results depicts robustness against selfish misbehavior.

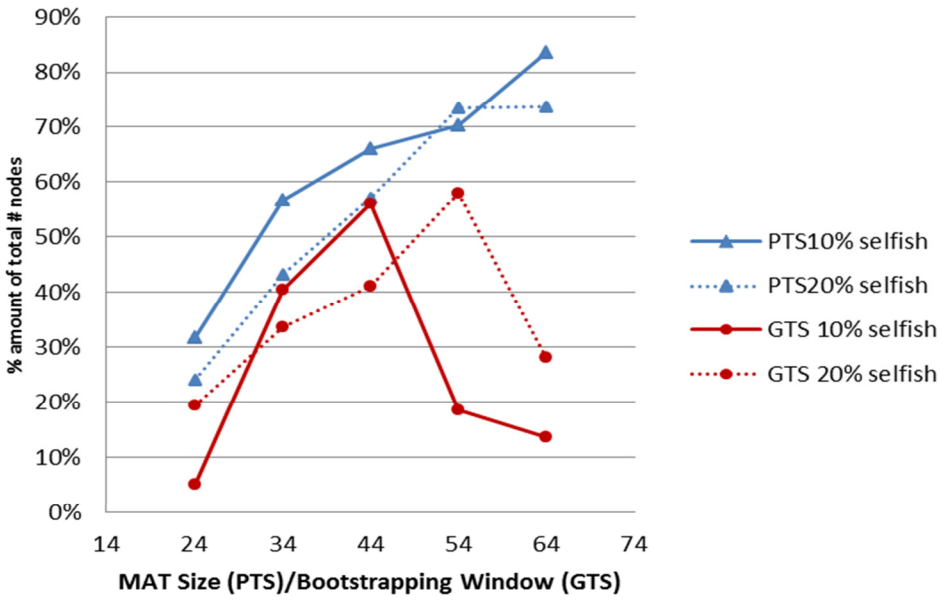


Figure 5.18 Nodes selection usefulness for different MAT values.

Figure 5.18 illustrates amount of useful selection messages for PTMS, in comparison to GTMS for 10% and 20% selfish misbehavior. In the figure, we may observe that selection usefulness in PTMS is higher,

for both 10% and 20% selfish misbehavior, than that of GTMS. Results are as expected showing higher selection usefulness when compared to GTMS. In addition, usefulness of PTMS increase when increasing MAT values, unlike GTMS which reduces after 44 bootstrapping window value for 10% selfish misbehavior, and after 64 bootstrapping window value for 20% misbehavior.

If we consider PTMS for 10% and 20% selfish misbehavior we may observe that 10% achieved more usefulness than 20% selfish misbehavior. This is due to the low amount of misbehavior in 10% experiments. Now, if we remember that a misbehaving node cheats with probability value of 0.25. Then we may understand that increasing amount of misbehaving services results in more un- useful service alternative selection messages.

If we consider the GTMS results we may find that 10% selfish misbehavior usefulness results reached it maximum at 44 Bootstrapping Window while at 20% reached its maximum usefulness at 54 Bootstrapping Window value. This bell curved-like result is normal, where the feedback system utilized in GTMS reaches to its limit point after which starts to produce un-useful selections. After 44 bootstrapping widow size the system consumes larger number of messages to reach atifaction resulting in reduction in amount of satisfying messages to ttoal number of selecttio messages.

5.2.10.1.1. Confidence Interval

In order to have confidence in our experiments' results, we have investigated the simulation time window to see if our simulations produced suitable results. Towards that, we did 3 groups of simulation each of 5 runs for 64 MAT and 10% selfish misbehavior. In the first 5, we used 0.75 of simulation time only, in the second we used the same time while in last we used twice simulation time. Given that, we've performed the paired t-test to compare the three group of runs with our work presented in this dissertation:

- our simulation results and the first 5 runs (0.75 simulation time),
- our simulation results and the second 5 runs (same simulation time),
- our simulation results and the third 5 runs (twice simulation time)

The paired t-test showed no significant difference between our simulations and the 0.75 simulation time and the same simulation time runs. The only significance was observed with twice simulation time.

Additionally, we have calculated the confidence interval range from our selection usefulness results given 64 MAT values and 10% selfish misbehavior. We found that for 95% confidence our satisfaction usefulness lower bound would be 74.6%, and the upper bound would be 92% . According to these upper

bounds we may conclude that our satisfaction results are at its lowest values would still outperform that of the GTMS results.

5.2.10.1.2. Comparing Distance Measures

The presented results showed that PTMS achived: higher saisfaction rates and low average number of messages, sent and receievd to/from PTMS server. In order to furtur evaluate our system and our results, we have chosesn to coduct number of experiments using different similarity measures other than the Eucledean distance for 64 MAT values for 10% selfis misbehavior. In our experiments we utilize euclidean distance to measure the degree of similarity between user's trust preferences and each service alternative's anticipated quality outcomes (evaluation metrics values). We've followed the evaluation study represented in [51], which proposed a technical framework to analyze, compare and benchmark the influence of different similarity measures on the results of distance-based clustering. Accordingly, we have selected two distance measures which are evalauted among the highest in terms of accuracy, convergence and performance , namely: Average and Manhatten distance.

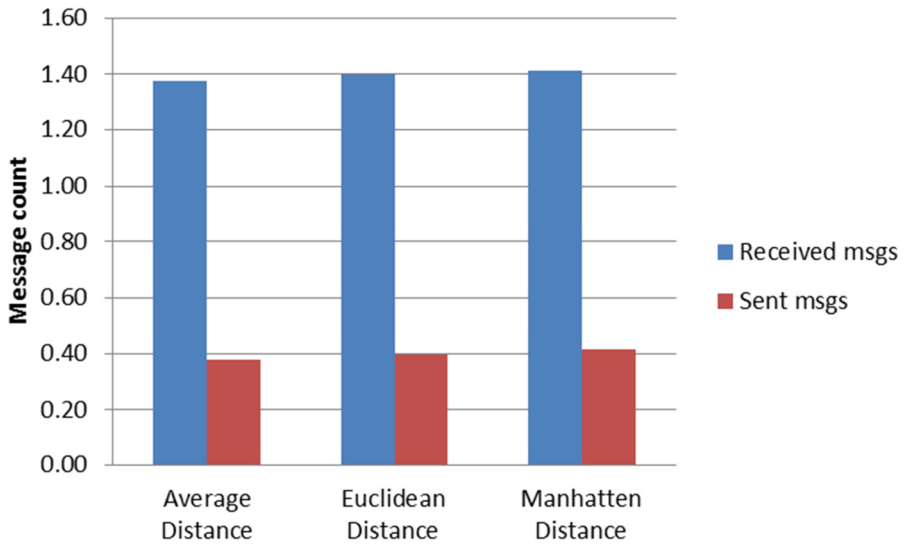


Figure 5.19 Message transaction ratio for different distance measures given 64 MAT and 10% selfish nodes.

We contrasted our results aganist that of the two distance measures, where we performed additional runs using averahe distance given 64 MAT and 10% selfish misbeahvior and then another 5 runs using

Manhatten distance for 64 MAT and 10% selfish misbehavior. We computed avergae distance as the following:

$$d_{ave} = \left(\frac{1}{n}\sum_{i=1}^{n}(x_i - y_i)^2\right)^{\frac{1}{2}}. \quad (5.14)$$

While computed manhatten distance as the following:

$$d_{man} = \sum_{i=1}^{n}|x_i - y_i|. \quad (5.15)$$

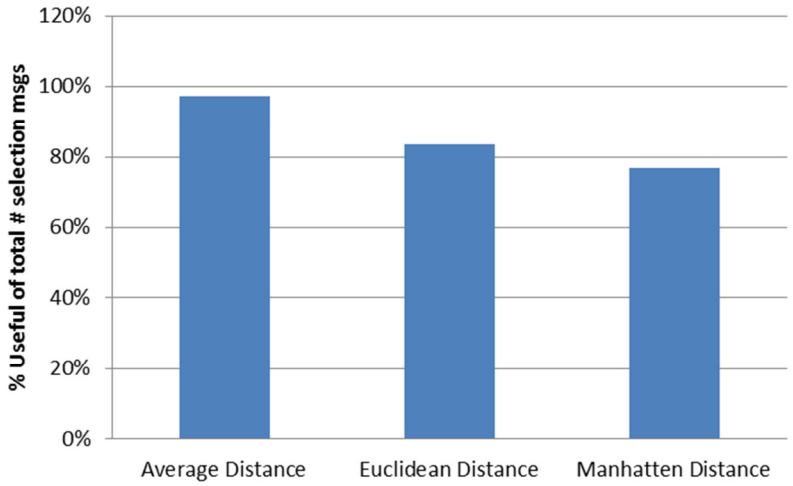


Figure 5.20 Selection usefulness for different distance measures given 64 MAT and 10% selfish nodes.

Figure 5.19 and 5.20 depicts results of the experiments. Figure 5.19, shows average number f messages generating per node that are sent and received by PTMS server. We can see that both Eucliedean and Manhatten disatnce measures prodced same number of messages, while Average distance measure produced fewer amount. This slight difference is due to the high satisfaction of nodes resulting in less number of messages sent and received across the network.

Figure 5.20 illustrates amount of useful service alternative selection messages. In the figure we can view that the Average distance measure, enhanced the selection process resulting in 97.2% of the slection messages were useful. This enahncement is due to the fact that average distance is not senstive to large value sof metrics undlike eucledean diatance and manhaten distanc.

5.2.10.1.3. PTMS without Clustering

For further evaluation of PTMS, we have compared the results for 64 MAT at 10% selfish misbehavior against that of a PTMS version without clustering operation. In other words our system stores collected data by MM at DM and uses it in its reasoning operations. Basically, the modified PTMS when receiving user's trust preferences computes the average value for all service alternatives and then searches for the closest service alternative to user's trust preferences. We utilized Euclidean distance for distance measure. We have compared message transaction ratio for sent and received messages and selection usefulness.

Figure 5.21 Message transaction ratio for PTMS with and without clustering operation.

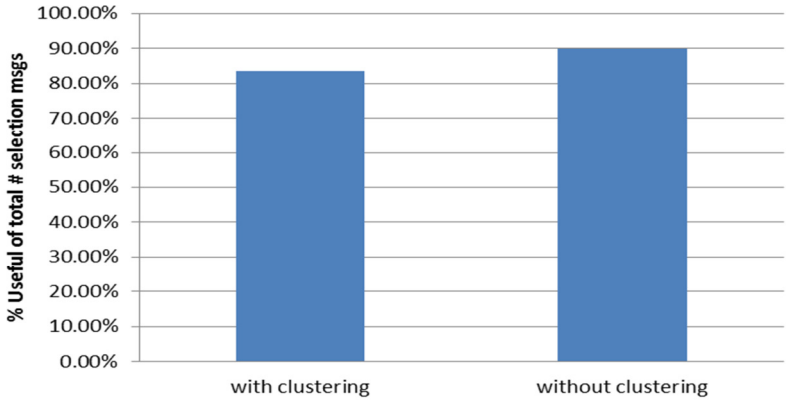


Figure 5.22 Selection usefulness for PTMS with and without clustering operation.

Figure 5.21 and 5.22 depicts our results. Figure 5.21, illustrates message transaction ratio for sent and received messages. We can see in the figure the huge different in average number of messages. This is expected as we utilize clustering to reduce amount of data stored in our system.

Figure 5.22, illustrates selection usefulness for PTMS with and without clustering. We can see from the figure an increase in selection usefulness of 5.3% when disabling the clustering operation. This is expected as we have on loss of data during data normalization operation and clustering operation.

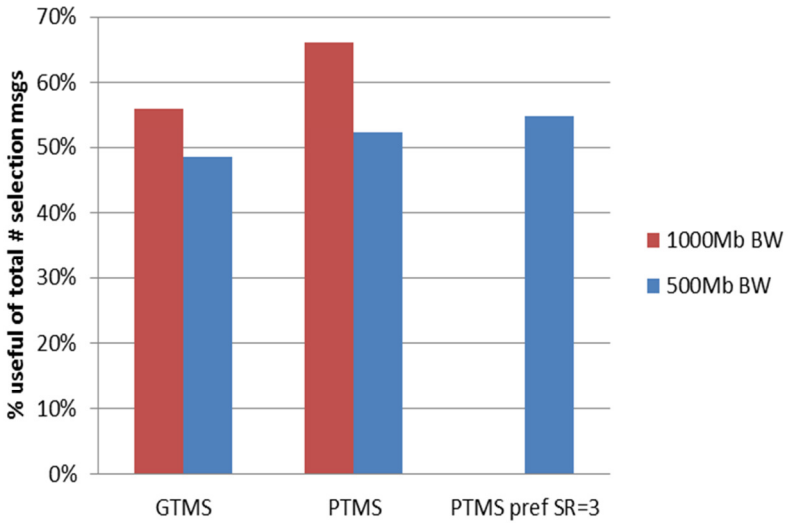


Figure 5.23 Selection usefulness given 1000Mb and 500Mb bandwidth for 44 MAT and 10% selfish misbehavior.

5.2.10.1.4. Changing Context

To complete our evaluation, we investigated the effect of changing context on PTMS and GTMS selection usefulness. We utilized half the network bandwidth of our previous experiments to be 500Mb. Additionally we tried to demonstrate the ability of the system to adapt according to changes in context and thus produce better results. This was achieved by allowing parties' to modify their preferences such that they receive better network management services. We conducted 5 separate runs using: GTMS, PTMS, PTMS having all users trust send rate preferences set to the value 3. Then we contrasted our selection usefulness results against that of our previous results (i.e. selection usefulness given 1000Mb). Our experiments were performed given 44 MAT value and 10% selfish nodes. We choose these settings as they recorded highest satisfaction values by GTMS. Figure 5.23 depicts the results. In the figure, if we

compare GTMS and PTMS results, we may observe a reduction in selection usefulness when decreasing available bandwidth for both systems. This is caused by packet loss and congestions occurred in the network. However, if we consider the difference between GTMS and PTMS selection usefulness, we may find that the gap between them has been affected, resulting in very close values in case of 500Mb. This is caused by users trust preferences, which resulted in unsuitable service alternative selections given the degree of congestion and packet loss found the network. This is illustrated when we increased sending rate in all users trust preferences, demonstrating the ability of the system to adapt when changing the context.

5.2.10.1.5. Diversities in Trust Preferences

As we can see from our previous experiments, the PTMS achieved higher effectiveness and efficiency values than the generic trust system. If we consider the ratio between number of trustors' unique preferences and number of service alternatives serving them, we may find that it is equal to 1.5.

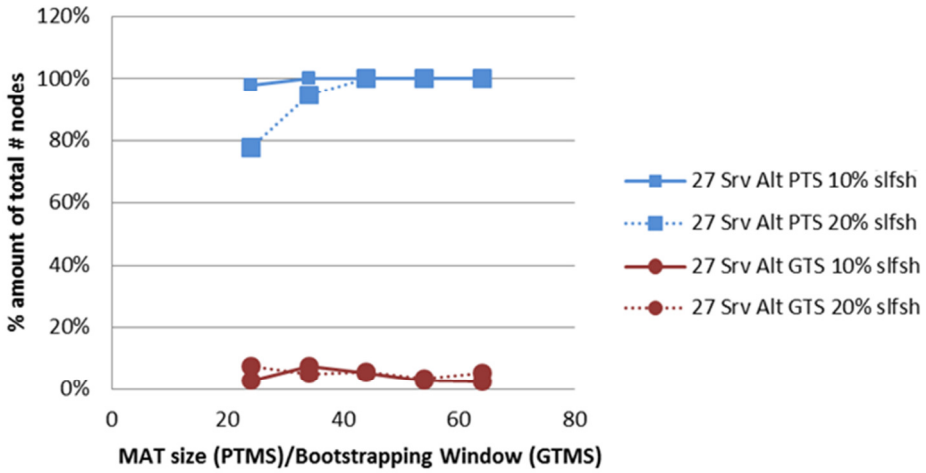


Figure 5.24 Nodes selection usefulness for different MAT values, given 27 service alternatives.

In other words, each service alternative approximately may satisfy 1.5 nodes across the simulated network. Such ratio constitutes extremely high diversity in trustor nodes preferences. Additionally, in our BitTorrent case study, if we consider trust preferences diversity we may find that we have 30 consumers of the file (trustor nodes) served by 30 providers (service alternatives) which also constitutes extremely high trust preferences diversity.

In order to confirm our results, we have chosen to consider two more cases of trustor nodes diversities, including; medium and low diversities. In the experiments, we used two populations for medium and low diversities. The first and the second are, 100 trustor nodes and 27 service alternatives, and 100 trustor nodes and 4 service alternatives respectively. The first population diversity of trustor nodes preferences is 3.7, while the second is 25. Towards that we have conducted two more sets of experiments to measure selection usefulness (Effectiveness) of PTMS and GTMS.

Figure 5.24 illustrates our results for medium nodes preferences diversity, and figure 5.25 illustrates our results for low nodes preferences diversity. In the two figures, we may observe that for both medium and low diversities, the PTMS achieved maximum usefulness (100%) value. Here, the max usefulness results are caused by the reduction in diversity of nodes preferences, resulting in less overhead endured by in the system. Such low overhead increased correctness of choice resulting in higher count of messages causing satisfaction of nodes.

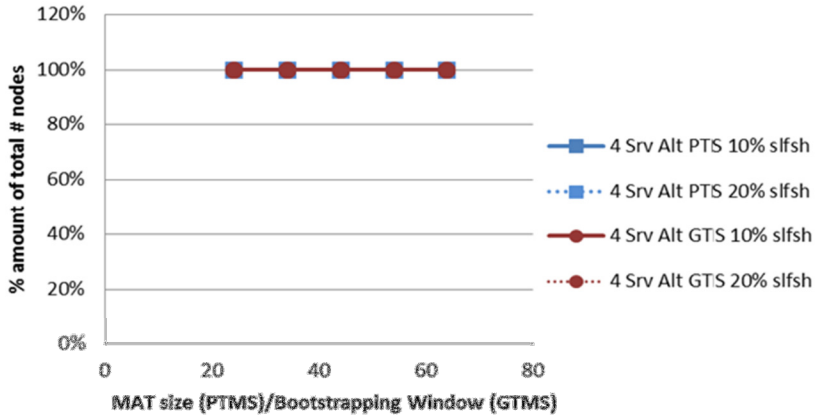


Figure 5.25 Nodes selection usefulness for different MAT values, given 4 service alternatives.

On the other hand, GTMS recorded low usefulness values in case of medium diversity and achieved maximum usefulness in case of low diversity. The low usefulness values show that the 27 service alternatives still constitute high diversity for the GTMS, which results in the system consuming larger number of selection messages that do not cause satisfaction of nodes. The maximum values achieved are caused by the very low number of service alternatives resulting in successfulness of voting process among nodes.

5.2.10.1.6. Scalability

As a part of our evaluations, we conducted group of experiments to demonstrate PTMS's scalability. In these experiments we've increased number of nodes to reach 200 and 300, under 64 MAT value and 20% of service alternatives exhibiting selfish misbehavior. Particularly, we choose 20% selfish misbehavior, as it is the case where the highest amount of data is generated. We experimented using low, medium and high diversities and contrasted the results against that of our simulations for 100 nodes population. Figure 5.26 and 5.28 depicts our results. Figure 5.26 represents total amount of messages generated by the system for 100, 200 and then 300 nodes given low, medium and high diversities. In the case of high diversity, we can see a linear increase in number of messages sent and received by the system. The factor of increase rate is 1.233. We can see the same increase for low and medium diversities; however the factor is 1.158 in the case of medium diversity and 1.163 in the case of low diversity. The linear increase observed in all cases is caused by the high level of satisfaction of nodes in the network, where nodes if not satisfied contacts the trust management server via selection request message and accordingly receives reply from server. In this fashion nodes keep sending and receiving message till reaching satisfaction.

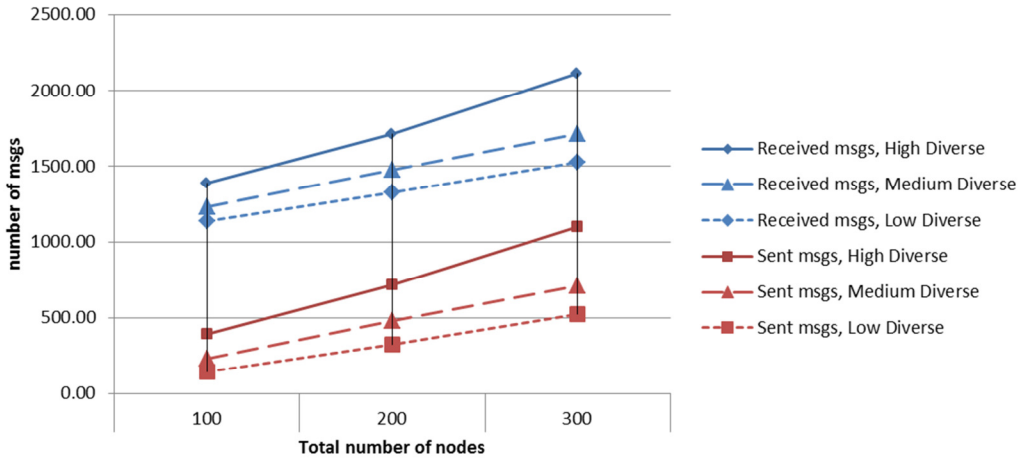


Figure 5.26 Total number of messages generated by the system for different populations, for low, medium and high diversities.

Figure 5.27 illustrates average number of messages generated by each node in the system. We can see the same linear increase observed in the three cases as figure 5.26.

Figure 5.28 depicts amount of used data when having 100, 200 and 300 nodes given low, medium and high diversities. We can see in the figure, for high diversities, that the used data is almost constant (8.9%) for 100 and 200 nodes. While decreases to 8.1% when increasing number of nodes to 300. The increase is insignificantly low; however it is caused by the increase in number of similar patterns in the network resulting in less number of cluster heads generated by the system.

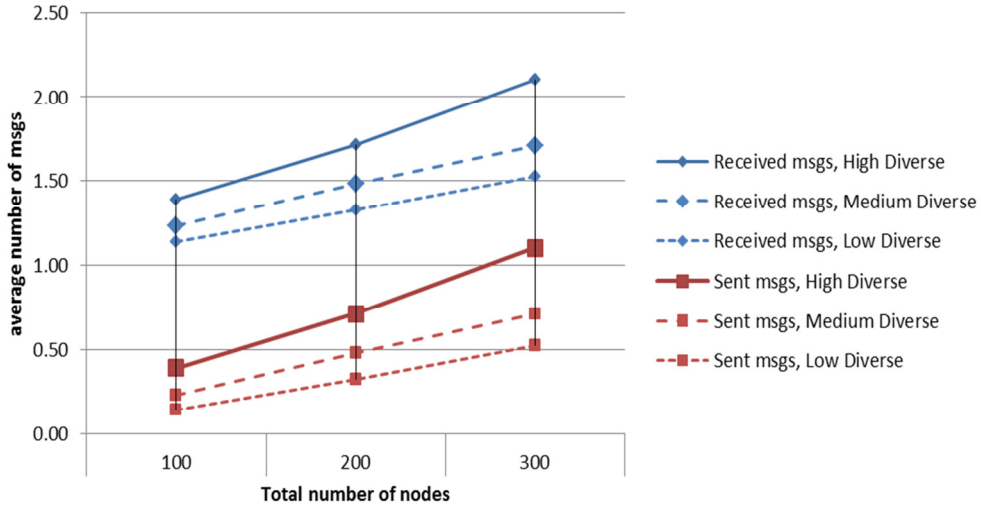


Figure 5.27 Message transaction ration for different populations for low, medium and high diversities.

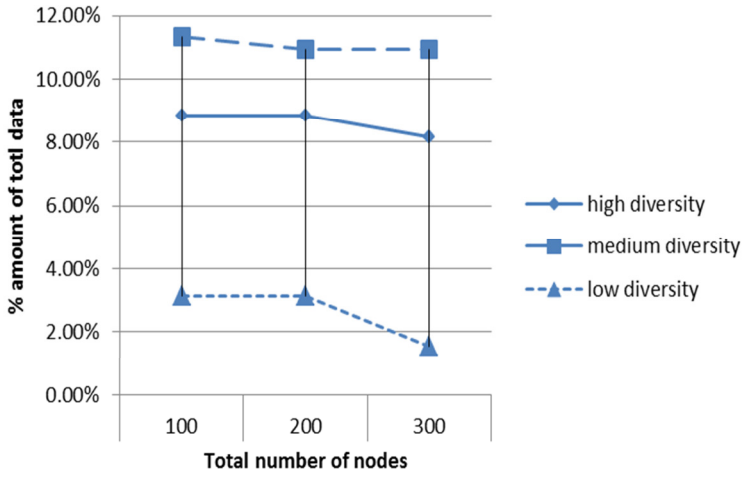


Figure 5.28 Amount of used data for different populations for low, medium and high diversities.

If we consider the case of medium diversity, we may find constant storage amount depicting higher values than that of high diversity. This is caused by increasing number of nodes and reducing number of service alternatives. In such situation, selfish services have higher probability of cheating than the case of 100 nodes, resulting in more patterns produced in the collected data.

Finally if we consider the low diversity case we may find constant amount of storage at 100 and 200 nodes and a linear decrease at 300 nodes. This linear decrease is caused by having more common patters of data when increasing number of nodes.

5.2.10.1.7. Findings

From our previous results we may conclude that,

- Selection usefulness increases when increasing history window size.
- Results recorded by PTMS for 10% and 20% selfish misbehavior, shows very close values depicting resilience of system towards the selfish behavior.
- Stored data (clustering data model) by the system decreases when increasing history window size due to the increase in common patterns among data points.
- Stored data (clustering data model) by the system decreases when increasing number of nodes for low medium and high diversities due to the increase in common patterns among the collected data.
- Number of messages generated in the network by the system increases linearly when increasing number of nodes.
- Utilizing Average, Euclidean and Manhattan distance for selecting most suitable service alternative showed high usefulness in case of average distance, medium and low in case of Euclidean and Manhattan distances respectively.
- Number of messages created in the network by PTMS reduces when increasing history window size.
- Disabling the clustering process decreases selection usefulness and considerably increases the amount of messages in the network.
- In case of low diversities among nodes preferences, a good choice is to utilize a GTMS system. Such system, in this case, would achieve the same high usefulness values of PTMS, while at the same time would preserve low processing and data complexities.
- In the case of high and medium diversities among nodes preferences, the PTMS should be utilized to achieve better effectiveness than GTMS.

- On the contrary of GTMS, when applying changes to the context, PTMS may adapt in order to sustain its high selection usefulness value.

5.2.10.2. Flexibility of PTMS

In this section we illustrate our experiments to verify and demonstrate flexibility of PTMS.

5.2.10.2.1. Experiment Setup

In this section we illustrate our experiments to verify and demonstrate flexibility of PTMS. Towards that, we utilized two trust computations on a single PTMS server, and divided the requirements of users trust in the network among the two trust computations. Trust computation is performed by the operations of DS, EM and AN components of RATM. In simulation, we had 50% of users trust having their preferences in terms of, Intent, Integrity and Results (IIR) metrics (section 4.1), while the other 50% having their trust preferences in terms of, SR, C and R metrics (SRCR). Both IIR and SRCR values were computed by PTMS using a single trust data model generated according to the same procedure used in our previous experiments (section 5.2.3). In other words, we utilize a single data model in which, MM operates on collecting its raw values, DS stores it, while EM and AN cooperates to generate it using the same procedure utilized in our previous experiments.

We followed the same simulation scenario of our previous experiments as in section 5.2.2.4.

IIR values were computed by PTMS as follows,

Intent metric was calculated as the diffusion value of service alternative. This was done by counting number of nodes in trust data model who utilized the service alternative.

Integrity metric was calculated as the deviation of the outcome of a service alternative from what was advertised by the service. This was done by implementing an average feedback score, in which the value 1 represents the service alternative delivering as advertised and the value 0 otherwise. We implemented an individual feedback score for each of the metrics, SR, C and R.

Results metric was calculated as the average score of what was delivered by the service alternative. We used three average scores for each of the metrics, SR, C and R.

Figure 5.29 illustrates a sample of used trust preferences in simulation.

We included the same misbehavior model of our previous experiments (section 5.2.2.1), and also used 10% and 20% selfish misbehavior amount. Our evaluation metrics were, Message-transaction ratio, Selection usefulness and Selection Accuracy of PTMS server. Selection accuracy is measured as the

Euclidian distance between utilized service alternative outcome and user trust preferences. Message-transaction ratio and Selection usefulness are measured as in our previous experiments.

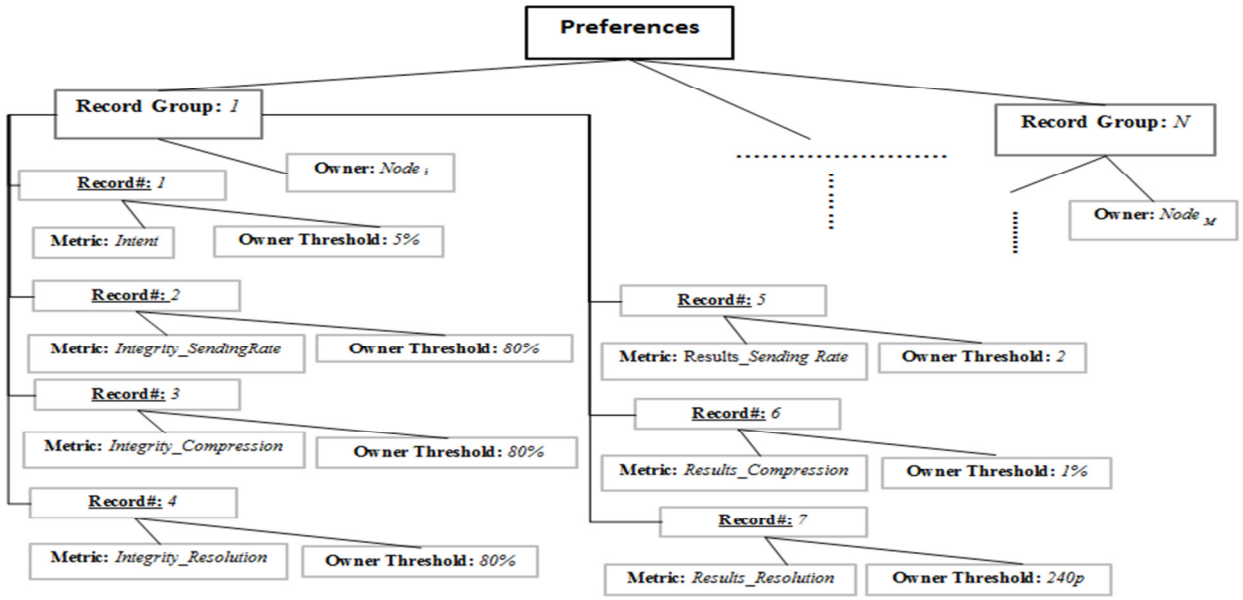


Figure 5.29 Sample trust preferences used in simulation.

5.2.10.2.2. Results

Figure 5.30 to 5.33 illustrates our results. Figure 5.30 illustrates message-transaction ratio for received messages (In-msgs) by the PTMS server, for MAT values, 24 to 64. The figure shows results for 10% and 20% selfish misbehavior for SRCR and IIR. In the figure we may observe that IIR metrics produced higher number of messages than that of SRCR. This shows that in IIR experiments, users trust sent larger number of messages to reach satisfaction state than that in the case of SRCR. Therefore, in average, satisfaction of nodes using IIR was reached at a point in simulation time after SCRC did.

In the figure may also observe a decrease in message count while increasing MAT values. This is due to the better selections achieved when increasing history window size (MAT). Finally, the least amount of messages being received is recorded at 64 MAT.

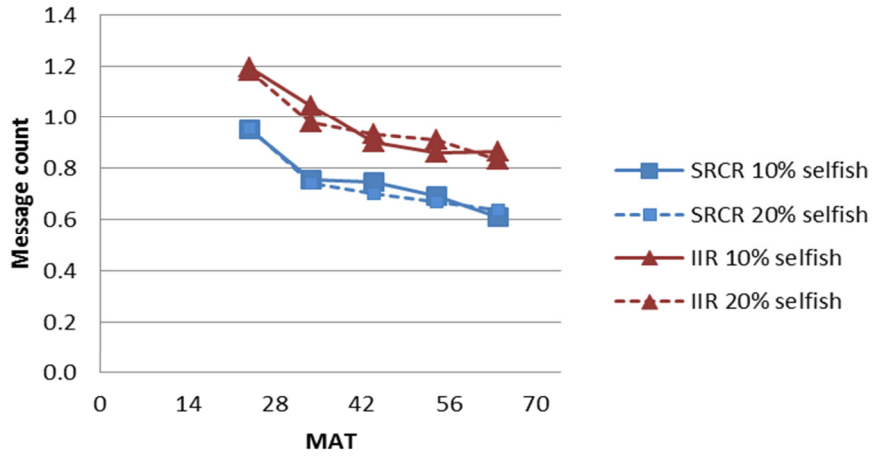


Figure 5.30 Message-transaction ratio for received messages (In-msgs), for different MAT values at 10% and 20% of nodes exhibiting selfish behavior, given 50% of nodes' preferences, Sending Rate, Compression and Resolution metrics and the other 50% preferences, intent, integrity and results metrics.

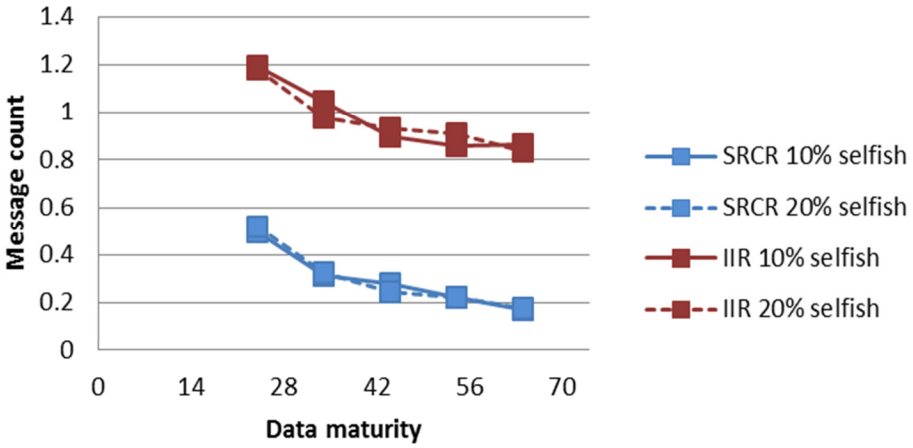


Figure 5.31 Message-transaction ratio for sent messages (Out-msgs), for different MAT values at 10% and 20% of nodes exhibiting selfish behavior, given 50% of nodes' preferences, Sending Rate, Compression and Resolution metrics and the other 50% preferences, intent, integrity and results metrics.

Figure 5.31 illustrates message-transaction ratio for sent messages (Out-msgs) from PTMS server, for MAT values, 24 to 64. The figure shows results for 10% and 20% selfish misbehavior for SRCR and IIR. In the figure, we may observe that IIR metrics produced higher number of messages than that of SRCR. This difference is due to the fact that, in the case of IIR, each monitoring message sent by a node to the PTMS server, a reply message is being sent back in order to inform the sender of the current IIR status of the service. We implemented this additional reply message because in the case of IIR, the user trust node

cannot measure the current IIR values of the service alternative as they target the global view about the service in the network. Here, this is unlike SRCR where the nodes can directly measure the delivered SRCR values immediately after each transaction. In figure we may also observe a decrease in message count while increasing MAT values. Again this is due to the better selections achieved when increasing history size. Finally the least amount of messages being sent is recorded at 64 MAT.

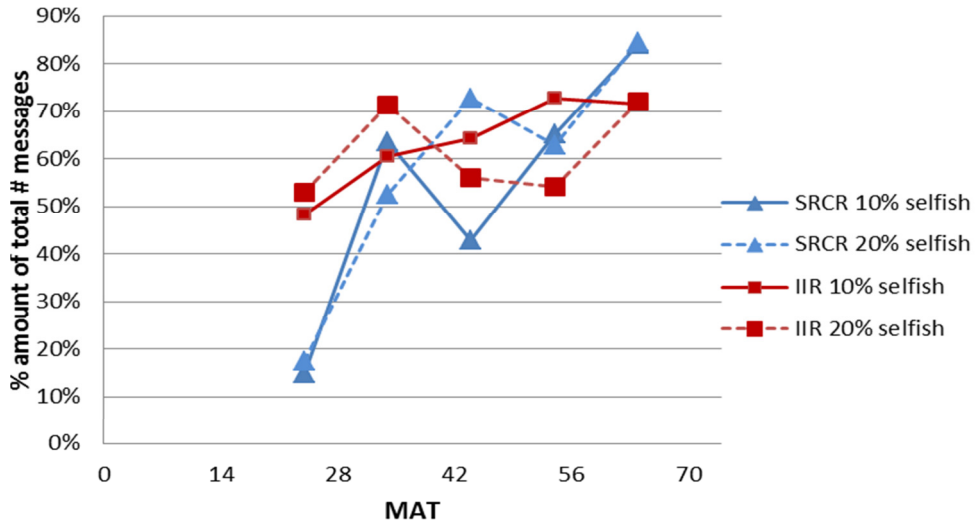


Figure 5.32 PTMS selection usefulness for different MAT values at 10% and 20% of nodes exhibiting selfish behavior, given 50% of nodes' preferences, Sending Rate, Compression and Resolution metrics and the other 50% preferences, intent, integrity and results metrics.

Figure 5.32 illustrates usefulness of PTMS server selection messages, for MAT values, 24 to 64. The figure shows results for 10% and 20% selfish misbehavior for SRCR and IIR. In the figure, we may observe that usefulness values tend to increase gradually while increasing MAT values. This increase is due to increasing MAT size, resulting in more data passed to the system and thus better decisions are made.

In the figure, we may observe oscillations between all recorded values. These oscillations tend to decrease when increasing MAT. Here the oscillations are caused by the insufficient amount of information used to generate the trust data model. Additionally oscillation values tend to be more visible in our results calculations when reducing number of nodes utilizing trust computation model (50% of total population).

In the figure also we may find that the difference between different usefulness points for the same MAT value tends to decrease when increasing MAT.

Finally at 64 MAT we may find that highest usefulness values and stable values where SRCR 10% and 20% record same results, while IIR does the same.

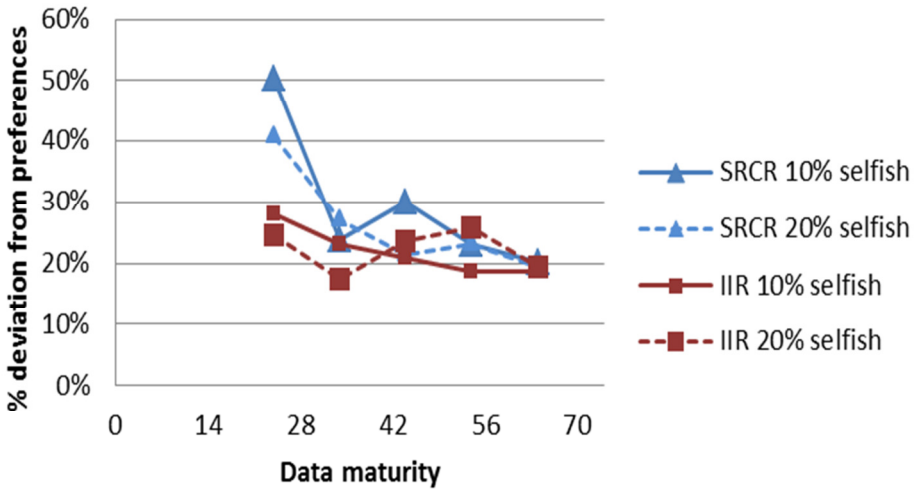


Figure 5.33 Selection Accuracy of PTMS for different MAT values at 10% and 20% of nodes exhibiting selfish behavior, given 50% of nodes' preferences, Sending Rate, Compression and Resolution metrics and the other 50% preferences, intent, integrity and results metrics.

Figure 5.33 illustrates accuracy of PTMS selections, for MAT values, 24 to 94, given 10% and 20% malicious behavior in the network. This was calculated as the Euclidean distance between service outcome and user trust preferences. In the figure, we may observe that the error in selections decreases with the increase of MAT till producing the same error value at 64 MAT. Such decrease is due to the increase in information size passed to the data model which results in better decisions made by PTMS.

In the figure also, we may observe the same oscillations as in figure 5.32.

Again it depicts same variations in values and reduces while increasing MAT values.

5.2.10.2.3. Findings

Form the previous results we may conclude the following,

- Differences between selection usefulness values for the two trust computations given 10% and 20% selfish misbehavior, decreases dramatically while increasing history window size (MAT). The close values in addition to the same selection error depict flexibility and effectiveness of PTMS.
- PTMS selection usefulness increases with the increase of history window size (MAT).
- Utilization of Intent and Integrity trust metrics enhanced selection usefulness in cases of low history values.
- Selection error of PTMS is reduced when increasing history window size (MAT).
- Sent and received message count both decrease when increasing history window size (MAT).

5.3. When to Choose PTMS

The main aim of our discussion in this section is to identify when to utilize a PTMS. Our experiments showed that PTMS produces better effectiveness and efficiency metrics values given high and medium diversities of trust preferences. Under low diversities, both systems produce equal results. An important question here is; what are the factors which governs our choice of selecting the suitable system? Should we select PTMS for, all cases, high and medium diversity cases or in high diversity cases?

From our PTMS presentation, we may clearly observe the overhead endured to provide the aforementioned effectiveness and efficiency values. This includes; clustering overhead, communication overhead and the distributed monitoring overhead.

In our implementation of PTMS, we utilized data clustering method in order to generate descriptive model of the collected data from the network. Our descriptive model is essential to reduce time and size complexities required for data in our system.

We deployed PTMS as a central server which provides trust management services in the network. Accordingly for the centralized PTMS model, our communication operations included sending and receiving, monitoring messages, service alternative selection messages. The monitoring messages included two types, namely, start transaction notification messages and collected data messages. The first informs monitoring component of start of a new transaction and thus orders MM to start monitoring operations. The second type sends the collected data from MM to the central server. The collected data included data sent from both the consumer side and the provider side. Here the size and count of messages matter. In our simulation, we used 3 metrics only and utilized 3 integers to hold all observed

values per transaction. However in real life that would depend on number and type of data used to express the monitored metrics.

In addition to monitoring messages, there exist service alternative selection messages. This type of messages count as observed from our results, increases with the low satisfaction and decreases with high satisfaction. Here, nodes keep requesting selection from the server till reaching to a satisfying condition.

Lastly, monitoring threads are distributed across nodes in the network. Overhead of these threads may vary, where some may be light weight, while others may require large memory and processing resources. This depends on the coded algorithm and how it monitors the required metrics.

From this discussion, we may understand the overhead endured by PTMS. Intuitively, it is high than GTMS, where GTMS receives feedback from users and applies summation or average operation only. Here we have no clustering, monitoring and messaging among system components. Such overhead endured by PTMS constitutes the cost we pay to achieve the desired effectiveness and efficiency values. Here, the degree of tolerance of nodes towards this overhead may provide insights on which system they may choose.

5.4. Conclusion

In this chapter we presented our evaluations of the proposed personalized trust management system. We utilized the proposed reference trust management architecture to design a personalized and a generic trust management system. Our approach, for evaluation was based on contrasting effectiveness and efficiency of both systems for two case studies, namely, BitTorrent file sharing and video conferencing application. For BitTorrent case study and given high trust preferences diversity, our results showed that,

- The personalized system produced better effectiveness and efficiency metrics values.
- Trust personalization increased resiliency to selfish poisoning attacks.

For video conferencing application case study, we experimented for three different trust preferences diversities, namely, low, medium and high. In addition we evaluates the proposed personalized system's scalability. Our results showed that,

- For high and medium diversities, the personalized system produced better effectiveness, efficiency and resilience results.
- For low diversities, the personalized system produced same results as the un-personalized system.
- For low medium and high diversity, PTMS showed: linear increase in number of messages and a reduced amount of stored data when increasing total number of nodes in the network.

Additionally we demonstrated flexibility of our solution, by implementing two different trust computations in our personalized system. The first trust computation defined and quantified trust in terms of, Intent, Integrity and Results parameters. While the second defined and quantified trust using simple network application parameters Our results showed,

- Close effectiveness, efficiency and resilience values among the two trust computations
- The utilization of the human inspired trust computation parameters, Intent and Integrity enhanced efficiency and resilience given small history window size.

Chapter 6

6. Conclusion and Future Work

6.1. Conclusion

In this work, we proposed RATM, reference architecture for trust management. The proposed reference architecture applies separation of concern among trust management operations: decision expectation, analytics, data management and monitoring. It defines trust management operations through five reconfigurable components which collectively can be used to implement a wide spectrum of trust management systems ranging from generic to highly personalized. We used RATM for trust personalization, where we proposed a Personalized Trust Management System (PTMS) based on RATM. We evaluated PTMS's effectiveness, efficiency and resilience and compared them to that of a Generic Trust Management System (GTMS). We used two case studies for our evaluations: BitTorrent and video conferencing applications.

In the BitTorrent case study, we contrasted a simple version of PTMS which adapts according to the preferences of peers in the network against a number of simple GTMSs, each with fixed thresholds matching peers' preferences in the network. In the video conferencing case study, we contrasted a more sophisticated version of our proposed PTMS against a GTMS which selects average service alternative in network based on trustors feedbacks. In the first case study, we investigate effectiveness, efficiency and resilience given high trust preferences diversities, while in video conferencing case study we investigated effectiveness, efficiency and resilience given low, medium and high trust preferences diversity. Our findings demonstrated flexibility and scalability of PTMS and showed that:

- For low diversities, PTMS produced the same results as GTMS.
- For high and medium diversities, PTMS produced better effectiveness, efficiency and resilience results.
- The utilization of the human-inspired trust computation parameters of Intent and Integrity enhanced efficiency and resilience given small history window size.

Intellectual Merit

In this work we propose RATM, reference architecture for trust management. RATM comprises a number of reconfigurable components that can be used to implement a wide variety of trust management system ranging from traditional systems (un-personalized) to personalized systems. We assume service-based environment where consumers, brokers and service providers are interacting and transacting in services and resources to meet their own trust requirements. We used the reference architecture to implement a personalized trust management system. The main contributions of this work are as follows:

- Proposing RATM for the guidance and development of a wide spectrum of trust management systems, ranging from un-personalized to fully personalized systems:
    - RATM, decouples different trust management operations: decision, expectation, analytics, data management and monitoring.
    - RATM comprises a number of distributed reconfigurable components which can be configured to coordinate and communicate with its counterparts to perform trust management operations.
    - In addition to, effectiveness and efficiency, the proposed reference architecture targets resiliency against attacks, extensibility, and scalability.
    - RATM may provide insights for the study of trust and reputation management.
- Proposing fully distributed personalized scalable trust management system based on RATM.
- Defining and quantifying trust in terms of four "quantifiable" parameter sets denoting Intent, Integrity, Capability and Results.

We evaluated trust personalization using simulation, where we used RATM to implement personalized and generic trust management systems.

Our research findings are as follows:

1. Personalization produces better effectiveness, efficiency and resilience values than current approaches (un-personalized), given high and medium diversities of trust preferences.
2. Given low diversities of trust preferences, personalization produces the same results as the current approaches (un-personalized).
3. Personalization endures high time and space complexities when compared to current (un-personalized) approaches, hence a tradeoff is required when choosing among personalization and current approaches (un-personalized).

4. Trust data sampling using clustering techniques enhances scalability of the personalized system.
5. The utilization of the proposed human-inspired trust model trust computation enhanced effectiveness and resilience of the system.

Broader Impact

Trust is the key enabler of interactions, where we cannot socialize, seek advice, consult, cooperate, buy or sell goods and services from/to others unless we establish mutual trust with our interaction parties and our interactions. RATM provides a reference architecture for trust management which can be used to implement a wide spectrum of trust management systems ranging from personalized to un-personalized systems. Personalization, in general, paves the way for reaching high levels of satisfaction, where interaction party's requirements are individually considered and thus consumers are served the best suited service(s). Hence, we claim that PTMS would largely enhance large-scale heterogeneous systems offering services and resources. This leads to more cooperation, more transactions, more satisfaction, less malicious behavior and lower costs would occur.

In addition, RATM may be used for the design and implementation of trust management systems in large scale network environment, such as PlanetLab [41]. In this scope, the deployed trust management system and its application would sever as a real world test bed for studying trust and reputation management. In such environment, scenarios of real world may be studied and new forms of trust computations can be tested and evaluated.

6.2. Future Work

Our future work includes the following:

- Expand our study beyond establishing trust in services and resources to include activities. We promote the development of new collaborative trust management systems, where various trust management operations would involve collaborating entities. This includes extending our reference architecture to feature collaboration by utilizing two planes: the first holds collaboration policies for each of the five components of the reference architecture, while the second includes the five trust management operation components. In such a way, the proposed reference architecture can be reconfigured to make trust management operation components work individually or in groups to

produce desired outcomes. We prepared a concept paper which includes collaborative trust work and submitted it to IEEE International Conference on Collaboration and Internet Computing, 2017.

- Expand our study on the human-inspired trust computation model. This includes exploring and studying different functions which humans apply to the four parameters as well as the impact of different contexts and different applications on the functions applied to such a model.
- Optimization of trust personalization. Our results showed the effect of high, medium and low diversities of trust personalization on our evaluation results. The context of an application, in general, may have an effect on preferences diversity and thus may affect user's choice. Specific market places of our interest would include healthcare and business-to-business markets. Further research is warranted on malicious misbehavior and appropriate mitigation in such market places.
- Adaptation of trust personalization. For some cases, a trustor may not be aware of the best trust parameters, methods or even trust computations, which best suit his/her trust requirements. Therefore, it may be required from trust management system to learn and reason from different transactions to build and update its knowledge, according to which it selects and modifies different trustors' preferences. The main goal here of course would be to achieve trustors' maximum satisfaction.
- Expand our study on PTMS's effectiveness, efficiency and resiliency. This may include the deployment of our proposed personalized trust management system in a large scale network environment such as PlanetLab [41]. In this scope, our deployed trust management system application would serve as a real world test bed for studying trust and reputation management. We would use it to study scenarios of real world applications to our system and produce further evaluations. Additionally, we may conduct more elaborate evaluations of a fully distributed peer-to-peer implementation of PTMS for different clustering techniques, different distance measures, more sophisticated techniques for mitigating misbehaviors, and finally new forms of trust computations.

Publications

PAPERS

1) H. Salah and M. Eltoweissy, "Towards a personalized trust architecture", *Proceedings of Collaborative Computing: Networking, Applications and Worksharing (CollaborateCom), 2011*.

2) H. Salah and M. Eltoweissy, "Towards a personalized trust management system," *Proceedings of Innovations in Information Technology (IIT)*, 2012.

3) H. Salah and M. Eltoweissy, "PETRA: Personalized Trust Management Architecture", *Proceedings of in IEEE 17th International Conference on Information Reuse and Integration (IRI)*, 2016.

4) H. Salah and M. Eltoweissy, "On the Personalization of Trust Management", *Proceedings of 2nd Workshop on Bio-inspired Security, Trust, Assurance and Resilience (BioSTAR 2017) in conjunction with IEEE Security and Privacy*, 2017.

5) H. Salah and M. Eltoweissy, "Towards Collaborative Trust Management", *IEEE International Conference on Collaboration and Internet Computing*, 2017 .

POSTER

6) H. Salah, M. Eltoweissy and Ayman Abdel-Hamid, "Computational Trust for Peer-to-Peer Web Services", *Proceedings of the 16th IEEE International Conference on Network Protocols (ICNP 2008)*, 2008.

SUBMITTED PAPER

7) H. Salah and M. Eltoweissy, "Personalized Reconfigurable Trust Management", International Journal of Trust Management in Computing and Communications, 2017.

BIBLIOGRAPHY

[1] K. Aberer and Z. Despotovic, "Managing trust in a peer-2-peer information system," *Proceedings of the tenth international conference on Information and Knowledge Management*, CIKM '01, 2001.

[2] M. Gupta, P. Judge, and M. Ammar, "A reputation system for peer-to-peer networks," *Proceedings of the 13th international workshop on Network and Operating Systems Support for Digital Audio and Video*, NOSSDAV '03, 2003.

[3] Singh and Ling Liu. "Trustme: Anonymous management of trust relationships in decentralized p2p systems," Peer-to-Peer Computing, IEEE International Conference on, P2P'03 2003.

[4] S.D. Kamvar, M.T. Schlosser, and H. Garcia-Molina, "The Eigen trust algorithm for reputation management in p2p networks," *Proceedings of the 12th international conference on World Wide Web,* WWW '03, 2003.

[5] Li Xiong and Ling Liu, "Peertrust: Supporting reputation-based trust for peer-to-peer electronic communities," IEEE Transactions on Knowledge and Data Engineering, 2004.

[6] E. Michael Maximilien, "Towards Autonomic Web Services Trust and Selection," PhD dissertation, 2004.

[7] R. Sherwood, L. Seungjoon, and B. Bhattacharjee, "Cooperative peer groups in nice," Computer Networks: The International Journal of Computer and Telecommunications Networking - Management in peer-to-peer systems, Volume 50 Issue 4, 2006.

[8] T. Repantis and V. Kalogeraki, "Decentralized Trust Management for Ad-Hoc Peer-to-Peer Networks," *Proceedings of the 4th international workshop on Middleware for Pervasive and Ad-Hoc Computing*, MPAC, 2006.

[9] Priyanka Agrawal, "A Trust Framework for Autonomic Computing Systems," Master's Thesis 2006.

[10] J. O'Donovan, V. Evrim, "Personalizing Trust in Online Auctions," *Proceedings of the third starting AI researchers' symposium*, 2006.

[11] P. Shah and J. Pâris, "Incorporating trust in BitTorrent Protocol," International Symposium on Performance Evaluation of Computer and Telecommunication Systems, SPECTS, 2007.

[12] X.-W. Chu, X. Chen, K. Zhao, and J. Liu, "Reputation and Trust Management in Heterogeneous Peer-to-Peer Network," Springer Journal of Telecommunication Systems, 2009.

[13] F. Qin, J. Liu and L. Zheng, "Improving Robustness and Fairness on BitTorrent-Like P2P Systems," Communications and Networking in China, ChinaCOM, 2009.

[14] Tangpong and G. Kesidis, "A Simple reputation model for BitTorrent-like incentives," International conference on game theory for networks, GameNets'09, 2009.

[15] V. Merekoulias, V. Pouli, Y. Rebahi, S. Becker, K. Cabaj, G. Aristomenopoulos, S. Papavassiliou, "A Trust Management Architecture for Autonomic Future Internet," International workshop on Management of Emerging Networks and Services, 2010.

[16] D. Choi, Seunghun, Y. Lee and Y. Park, "Personalized Eigen Trust with the Beta Distribution," Electronics and Telecommunications Research Institute, ETRI journal, volume 32, number 2, 2010.

[17] W. Conner, A. lyengar, T. Mikalsen, I. Rouvellow and K. Nahrstedt, "A Trust Management Framework for Service-Oriented Environments," *Proceedings of the 18th international conference on World Wide Web*, WWW '09, 2009.

[18] G. Suryanarayana, M. Diallo and R.Taylor, ”A Generic Framework for Modeling Decentralized Reputation-based Trust Models,” Institute for Software Research, University of California, Irvine, ISR Technical report, 2007.

[19] M. Refaei, L. DaSilva, M. Eltoweissy and T. Nadeem, “Adaptation of Reputation Management Systems in Ad hoc Networks,” IEEE Transactions on Computers, Vol. 59, No. 5, 2010.

[20] Z.i Malik, “Reputation Management Framework for Service Oriented Environment,” PhD dissertation, 2008.

[21] J. Caverlee, L. Liu and S. Webb, “SocialTrust: Tamper-Resilient Trust Establishment in Online Communities,” *Proceedings of the 8th ACM/IEEE-CS joint conference on Digital libraries*, JCDL ‘08, 2008.

[22] S. Khan and K. Hamlen, “Hatman: Intra-cloud Trust Management for Hadoop,” IEEE 5th International Conference on Cloud Computing, 2012.

[23] X. Dai, C. Zhu, Y. Guo, “P2P dynamic trust management system based on trust network,” International Conference on Cyber Security, Cyber Warfare and Digital Forensic, CyberSec2012, 2012.

[24] I. Chen, F. Bao, and J. Guo, “Trust-based Service Management for Social Internet of Things Systems,” IEEE Transactions on Dependable and Secure Computing, 2015.

[25] X. Xie, Y. Li, ”Trust Management Model of Cloud Computing Based on Multi-agent,” International Conference on Network and Information Systems for Computers, CNIS2015, 2015.

[26] J. Lopez and S. Maag, “Towards a Generic Trust Management Framework using a Machine-Learning-Based Trust Model,” IEEE conference Trustcom/BigDataSE/ISPA, 2015.

[27] S. Roy, T. Sarker, and M. Hashem, ”A Novel Trust Measurement System for Cloud-based Marketplace,” *Proceedings of International Conference on Electrical Information and Communication Technology*, EICT, 2015.

[28] T. Noor, Q. Sheng, S. Dustdar,” CloudArmor: Supporting Reputation-Based Trust Management for Cloud Services,” IEEE Transactions on Parallel and Distributed Systems, Vol. 27, NO. 2, 2016.

[29] S. Covey and R. Merrill, “The Speed of Trust”, Book, published 2006.

[30] Tomh, MediaWiki, 2005, “ns-2 User Information,” Available http://nsnam.sourceforge.net/wiki/index.php/User_Information [Accessed June 2013].

[31] B. Frey and D. Dueck, “Clustering by Passing Messages Between Data Points,” in University of Toronto Science Journal of Science, Vol. 315, 2007.

[32] H. Salah, M. Eltoweissy and Ayman Abdel-Hamid, “Computational Trust for Peer-to-Peer Web Services,” *Proceedings of 16th IEEE International Conference on Network Protocols*, ICNP2008, 2008.

[33] H. Salah, M. Eltoweissy, “Towards a personalized trust architecture,” *Proceedings of international conference Collaborative Computing: Networking, Applications and Worksharing*, CollaborateCom, 2011.

[34] H. Salah and M. Eltoweissy, "Towards a personalized trust management system," *Proceedings of International conference Innovations in Information Technology*, IIT, 2012.

[35] H. Salah and M. Eltoweissy, “PETRA: Personalized Trust Management Architecture,” IEEE 17th International Conference on Information Reuse and Integration, IRI, 2016.

[36] PayPal, 2002, “PayPal Buyer Protection Policy,” available:www.paypalobjects.com/webstatic/en_AT/ua/pdf/buyerprotection.pdf [Acessed 20 July 2015].

[37] P. Resnick and R. Zeckhauser, "Trust among strangers in internet transactions: Empirical analysis of ebay's reputation system," *Proceedings of NBER workshop on empirical studies of electronic commerce*, 2000.

[38] Tutorials Point,2016, "Amazon MarktPlace," available: https://www.tutorialspoint.com/amazon_marketplace/amazon_marketplace_tutorial.pdf [Accessed 20 July 2015].

[39] G. Suvarna , Dr. Krishna, "Context Aware Service Information Extraction from Bigdata," International Journal of Computer Science and Information Technologies, Vol. 6 (1) , 2015.

[40] M. Kuhn, ISO and IEC, 1996, "Information technology-Syntactic Metalanguage-Extended ISO/IEC 14977:1996(E)," available: http://www.cl.cam.ac.uk/~mgk25/iso-14977.pdf [Accessed 20 September 2013].

[41] Andy, The Trustees of Princeton University, 2007, "About PlanetLab," available: https://www.planet-lab.org/about. [Accessed 1 July 2016].

[42] Bram Cohen," Incentives Build Robustness in BitTorrent, "Workshop on Economics of Peer-to-Peer Systems," 2003.

[43] D. Ilie, "Gnutella Network Traffc and Characteristics," Masters thesis in Telecommunication Systems, school of Engineering, Blekinge Institute of Technology, 2006.

[44] Hoi Chan, Alla Segal, Bill Arnold, and Ian Whalley," How Can We Trust an Autonomic System to Make the Best Decision?," *Proceedings of the second International Conference on Autonomic Computing*, ICAC, 2005.

[45] J. Lopez, S. Maag and G. Morales, " Behavior evaluation for trust management based on formal distributed network monitoring," Article in World Wide Web, 2015.

[46] H. JiuSong and H. Hong, "Reference Model of Trustworthy Proof for Trusted Components," *Proceedings of Second International Conference on Future Information Technology and Management Engineering*, 2009.

[47] D. Roman and G. Stefano, "Towards a Reference Architecture for Trusted Data Marketplaces," *Proceedings of 2nd International Conference on Open and Big Data*, 2016.

[48] L. Lacomme, Y. Demazeau, and V. Camps, "How to Integrate Personalization and Trust in an Agent Network," *Proceedings of International Conference on Agents and Artificial Intelligence ICAART*, 2009.

[49] J. Samson Mwela and O. Emmanuel Adebomi ,"Impact of Packet Loss on the Quality of Video Stream Transmission," Master Thesis in Electrical Engineering, School of Computing Blekinge Institute of Technology, 2010.

[50] L. McGarthwaite , "Client-Server versus Peer-to-Peer Architecture: Comparisons for Streaming Video," *Proceedings of the 5thWinona Computer Science Undergraduate Research Seminar*, 2005.

[51] A. Shirhorshid, S. Aghabozorgi and T. Wah, " A Comparison Study on Similarity and Dissimilarity Measures in Clustering Continious Data," research article published in PLOS ONE Journal, 2015.

[52] J. MacQueen, "Some methods for classification and analysis of multivariate observations," *Proceedings of Fifth Berkeley Symposium on Math Statist and Prob*, Vol. 1, 1967.

[53] X. Jin, J. Han, "K-Medoids Clustering," Encyclopedia of Machine Learning, Springer Link, 2011.

[54] E. M. Kriegel et al., "Knowledge Discovery in Large Spatial Databases: Focusing Techniques for efficient Class Identification", *Proceedings of 4th Int. Symposium on Large Spatial Databases*, 1995.

[55] P. Nagpal and P.Mann, "Comparative Study of Density based Clustering Algorithms," International Journal of Computer Applications, Volume 27– No.11, 2011.

'A' Appendix

Terminologies Used

Trust Computation: is the process of producing trust evaluations. This includes the operations applied to data in order to generate an evaluation.

Trust Data: all forms of data involved in trust computation.

Trust Architecture: is the trust management systems architecture.

Trust Programmability: is the degree of customizability of trust management system, in terms of producing trust evaluations which may meet different trust requirements.

Trust: is the belief or disbelief of a party that another party, for a said subject of trust (SoT) in a given context, has the ability to exhibit a set of acceptable actions in the future for the welfare of the trusting party.

Trustor: is the party who seeks and establishes trust. For instance: Alice trusts Bob. Alice is the trustor.

Trustee: is the party who is subjected to trust. For instance: Alice trusts Bob. Bob is the trustee.

Interaction Party: is a partner of another party in a transaction. For instance: Jack bought a book from Donald. To Donald, Jack is his interaction party. To Jack, Donald is his interaction party.

User Trust: is the party consuming the trust management service.

Service Consumer: is the party which seeks, requests and invokes service instances.

Service Provider: is the party which hosts or outsources the service instance in order for service requestors to invoke it.

Service Broker: is the party which considers service consumers' requirements and accordingly provides a list of service providers service pairs who can satisfy these requirements.

Service: is a set of actions to be performed by a service provider on one or more resources of his own, for the welfare of the service consumer.

Service instance: is an image of the service. For instance, the BitTorrent tracking service in Vuze file sharing software for The Dark Knight, Spiderman and The Avengers movies. Here, we have three tracking service instances, one per movie.

Resources: is whatever is needed to provide a service. It includes all hardware, software, skill set and raw material needed to provide a service.

Trust Preferences: is the set of requirements needed by a party to trust another party. For instance, for Alice to trust an email service, the email service has to be 90% available and 90% reliable. Availability and reliability and their threshold values constitute Alice's trust preferences.

Trust Profile: is the set of quality metrics describing a service. For example, an email service is 95% available and average 80% reliable. Both metrics and their values constitute Trust profile for the email service.

Trust Report: is the set of trust profiles of number of parties.

Quality Metric: is a computed value representing the anticipated future value of a metric. Such as average upload bandwidth at node A. Here the average value represents the anticipated future value of bandwidth at node A.

Intent: is the cognitive state representing willingness of a party to execute a given action

Integrity: is consistency in actions. It is mainly about delivering what is advertised and being consistent in delivery it to different parties.

Capability: constitutes all resources needed to deliver a service. These resources include hardware, software, skill set and raw material.

Results parameter: is the track of "promised or contracted" outcomes.

Leecher: a peer downloading and uploading chucks it downloaded so far from the BitTorrent file sharing swarm.

Diversities in trust preferences: is the ratio between the number of trustors' unique preferences and the number of service alternatives serving them.