Communications Resource Allocation: Feasibility Assessment for Tactical Networking Applications

Jon A. Bernard

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

Masters of Science
in
Computer Science and Applications

Dr. Richard E. Nance, Chair
Dr. James D. Arthur
Dr. Scott A. Midkiff

December 14, 2004
Blacksburg, Virginia

Keywords: Communications Resource Feasibility, Tactical Networks

Communications Resource Allocation: Feasibility Assessment for Tactical Networking Applications

Jon A. Bernard

(ABSTRACT)

The research reported here offers a solution to the communications resource allocation problem. Unlike earlier approaches to this problem, we employ a time-sliced event model where messages are sent and received in a single time slice called an epoch. In addition, we also consider networks that contain relay nodes capable of only transferring messages. Consequently, network topologies can be considered where a given node is not directly connected to every other node and must use one or more relay nodes in order to get a message to some destination. The resulting architectures broaden the networks to be considered and enable the capability of constructing more realistic communication scenarios.

In this paper we modify the standard MCNF model by turning our focus to feasibility instead of optimality in an effort to provide adequate and accurate decision support to communication network planners. Given a network configuration and message requirements, our goal is to determine if the proposed scenario is feasible in terms of the communication resources available.

To meet this goal, three algorithms are presented that each solve the extended MCNF problem with varying degrees of accuracy and run-time requirements.

Experimental results show that a large number of multi-variable interactions among input parameters play a key role in determining feasibility and predicting expected execution time. Several heuristics are presented that reduce run-time dramatically, in some cases by a factor of 37.

Each algorithm is tested on a range of inputs and compared to the others. Preliminary results gathered indicate that the second algorithm of the three (APEA) offers the best balance of accuracy vs. execution time.

In summary, the solutions presented here solve the resource allocation problem for message delivery in a way that enables evaluation of real world communication scenarios.

This work was supported and funded by the U.S. Navy and the Office of Naval Research as part of the NTNRADS program.

Acknowledgments

I would like to express my gratitude to my committee members: Dr. Nance, Dr. Arthur, and Dr. Midkiff. Thank you for your guidance and patience through the course of this research. The lessons learned from our research meetings and time spent together have proved invaluable to my professional career as well as my personal character. I could not have gained that from any course or textbook and I am truly grateful to have worked under professors of such high caliber.

A very special thanks goes out to my family, who's moral support and encouragement throughout my entire life has given me strength in times of difficulty. I could not have done this without you.

I must also acknowledge my friends in graduate school for their support and in particular, I must thank Anil Bazaz and Bharath Ramesh. You're two of the best friends I could have asked for.

Finally, I recognize that this research was made possible by the financial assistance provided by the Systems Research Center at Virginia Tech.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Many elements of our daily lives are influenced and affected by information technology. The same is true for our military organizations. This concept is made interesting by the fact that information technology is evolving and expanding at an astounding rate. Just as we adapt to this ongoing evolution, our military must constantly evaluate and improve strategies and methods used for waging war and conducting operations other than war, e.g. rescue of diplomatic personnel from a country in turmoil.

Naval strategies have seen dramatic changes in recent years. Prior to Desert Storm, warfare was dominated by a platform-centric paradigm where each fighting unit acted rather independently of other units. Communication was limited at best, especially in extreme battlefield conditions. Radio was the primary method of communication and was proved unreliable and subject to interference and interception. Transferring large quantities of information from one unit to another was not possible in many cases. The utility of information gathering and distribution was limited due to the available technology at that time.

After the advent of computer networks and the attendant explosion of communication network technology, the Navy began a fundamental shift in warfare paradigms from platform-centric to what is now called network-centric warfare [7]. The network-centric warfare concept, established by the rapid evolution in information technology, advocates the adoption of computer and communications based on newly available and more effective technology.

Accompanying these new communication capabilities is a whole new set of problems previously unconsidered. One such problem is determining how to construct a network to accommodate potential communications requirements. This is an integral part of the planning process and cannot afford to be overlooked. Proper and thorough planning is an essential ingredient to success on the battlefield. Gathering information is an important part of strategic planning, but even more important is the need for efficient processing of that information. The more intelligent opponent realizes a higher probability of success in a given battle.

The overall goal of obtaining and utilizing information is to increase intelligence and improve situational awareness on the battlefield. Efficient processing and transformation of data translates directly into the ability to make well informed decisions. A large part of the decision-making process involves resource allocation. Predicting where resources are needed most in a communications network is difficult, especially as a battle progresses. So, how much bandwidth does a planning team allocate to the primary communications network? In trying to answer this question the need for a resource allocation and decision support utility becomes apparent. A planning team must have the ability to determine if the current communication resources can accommodate the estimated requirements. In addition, the ability to evaluate the flexibility of a communication network is also highly desirable. If suddenly one fighting unit needs to upload a massive amount of data to a commanding unit, the planning team needs to know if the network can handle this kind of load. Until now, only preliminary progress has been made toward achieving this capability.

The research reported here offers a solution to the communication (bandwidth) resource allocation problem. Unlike earlier approaches to this problem [18], we employ a time-sliced event model where messages are sent and received in a single time slice called an epoch. In addition, we also consider networks that contain relay nodes capable of only transferring (relaying) messages. Consequently, network topologies can be considered where a given node is not directly connected to every other node and must use one or more relay nodes in order to get a message to some destination. The resulting architectures broaden the networks to be considered and enable the capability of constructing more realistic communication scenarios.

This paper is organized as follows. Chapter 2 provides a literature survey

that summarizes the prior work conducted in this area of research, forming a solid foundation for its importance. Chapter 3 explains the specific definition of the NTN:RADS problem and provides a mathematical formulation based on an extension of a well-known network flow problem. Chapter 4 gives an overview of the approach taken to develop the program, including run-time environment and how the input and output are structured. Chapter 5 details the implementation of each of the three algorithms and discusses the specific similarities and differences that exist. Chapter 6 focuses on the analysis of the program and provides comparative measurements of the different run-time options. A conclusion and discussion of potential research extensions is given in Chapter 7.

Chapter 2

Literature Survey

In this chapter we provide a survey on two areas of related research. We begin by examining the previous work on the communications resource allocation problem, providing a clear picture of the targeted application domain. Secondly, a summary of related network flow problems and various results is given. This section defines the mathematical foundation that forms the basis for a solution to the NTN:RADS problem.

2.1 Communications Resource Allocation

Naval tactical communications resource allocation was first examined by the Systems Research Center [3] research team in the Virtual Operations Network (VON) project. The goal was to combine the communication resources of partners of different countries and suggest an optimal communication configuration where all partners could communicate effectively. The most direct solution forces the transition of communications capabilities to the least advanced partner. Therefore, the solution accepts only limited capability and line-of-sight restrictions on communication networks.

Initial efforts sought to identify a data structure for communication tasks. Information exchange requirements (IERs) were recognized as meeting this purpose. Later the IER was further decomposed into message information exchange requirements (MIERs) that allowed a communications task to be

broken into individual transferable data units. The MIER was a key enabler for creation of a tool to evaluate communication requirements given a network topology and capabilities. The details of these efforts and subsequent results are described in two technical reports [15, 16].

Focus then turned towards constructing a model for resource allocation evaluation. An algorithm was designed and implemented as part of the tool used to provide decision support [18]. Subsequently, the research goals were expanded to meet the capabilities of the U.S. needs for C^{4}ISR[1] and focus now shifted from limited capability communication assessment to focus solely on national communication assessment. A new algorithm was required to accommodate the increase in expected capabilities. An existing algorithm by the name of NETFLO [14] was discovered, and necessary modifications were made to allow integration with the existing model. The distinctive function of NETFLO is the expansion to topologies having relay nodes. Now over-the-horizon distances could be considered, making the assessment model far more capable.

NETFLO, however, contained drawbacks not realized prior to integration into the assessment model. Although NETFLO provided support for relay nodes, it assumed that all messages being transmitted were identical. In the parlance of network flow modeling, each message represents a single commodity. This meant that NETFLO could not deliver unique messages to different locations. In essence, NETFLO could solve the *bandwidth allocation problem*, but could not solve the *message routing problem*.

Upon realizing this problem, the need for a new algorithm became apparent. This provided the motivation for the design and development of the algorithms reported in this research.

2.2 Background and Motivation

Until recently, naval tactical communications generally assumed a network would take on a line-of-sight restriction. In other words, each node can communicate directly with every other node. In an ideal world this would be

[1] C^{4}ISR stands for ***C**ommand, **C**ontrol, **C**ommunications, **C**omputers, **I**ntelligence, **S**urveillance and **R**econnaisance*

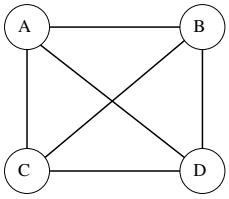

Figure 2.1: *Fully connected graph*

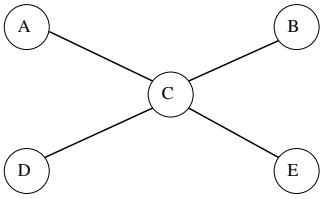

Figure 2.2: *Transshipment graph*

the preferred model. However, economically and technologically, this is no longer practical or acceptable. An example of such a network might appear as a fully connected graph, shown in Figure 2.1 In our research, we have discarded this assumption by allowing an over the horizon capability shown in Figure 2.2 as a transshipment graph. In this network nodes may not be able to communicate directly with other nodes but can use a relay node in order to transfer information to the desired destination. In this configuration node C in Figure 2.2 could be a satellite capable of relaying information and bridging the gap between nodes too far apart to communicate directly. The relay capability is a uniquely distinguishing feature of our research.

Tactical communications networks are also considered to be capacitated networks. In other words, each channel is capable of transmitting a finite amount of information in a single time unit. Once the capacity limit for each channel has been reached, no further information can be transmitted until the next time unit.

In addition to the network model, a mission plan is divided into time slices or epochs. Each epoch provides a discrete time unit during which events can occur. This framework provides an environment in which messages can have a time unit by which arrival is required. The notion of message due dates is also a unique feature of the research reported in this paper.

Given these requirements, our problem resolves to finding a routing schedule for a set of messages through a capacitated network where the parameters contained in each message define the time in which they can be sent. A successful routing schedule is one in which all messages arrive on or before their specified due date and no capacity on a single channel is exceeded.

The message routing problem (resource allocation) falls into a very broad category of network flow problems. Several different classifications of net-

work flow problems have been identified [4], each demonstrating particular characteristics and solution strategies. The resource allocation problem can be best described by a multicommodity network flow model, which is itself an extension of the minimum cost flow problem.

2.3 Network Flow Problems

The solutions described in this research are based on network flow models. This section provides a brief summary of related network flow problems and describes the fundamental components that all network flow problems have in common. This is followed by a description of the additional side constraints present in the NTN:RADS problem.

In the interest of clarity, the next two sections were found in [4] and modified only slightly in the process.

2.3.1 The Minimum Cost Flow Problem

The most basic of all capacitated network flow problems is the minimum cost flow model. The goal is to find the minimum cost routing schedule for a commodity withing a network. This routing schedule must ensure that demand and supply at each node is completely satisfied [4]. Formally, the problem is stated as follows:

Let $G = (N, A)$ be a directed network with a cost c_{ij} and capacity u_{ij} associated with every arc $(i, j) \in A$. We associate with each node $i \in N$ a number $b(i)$ which indicates its supply or demand depending on whether $b(i) > 0$ or $b(i) < 0$. The minimum cost flow problem can be stated as follows:

$$\text{Minimize} \qquad z(x) = \sum_{(i,j)\in A} c_{ij}x_{ij} \tag{2.1}$$

subject to

$$\sum_{\{j:(i,j)\in A\}} x_{ij} - \sum_{\{j:(j,i)\in A\}} x_{ji} = b(i) \qquad \text{for all } i \in N, \tag{2.2}$$

$$0 \leq x_{ij} \leq u_{ij} \qquad \text{for all } (i,j) \in A. \tag{2.3}$$

Let C denote the largest magnitude of any arc cost. Further, let U denote the largest magnitude of any supply/demand or finite arc capacity. We assume that the lower bounds I_{ij} on arc flows are all zero. We further make the following assumptions:

1. *All data (cost, supply/demand, and capacity) are integral.*

   This assumption does not need to be maintained in practice as we can always convert a rational number into an integer by multiplying it by a suitably large integer.

2. *The network is directed.*

   It has been shown that any undirected graph can be transformed into a directed graph [4]. Therefore we can always fulfill this assumption.

3. *The supplies/demands at the nodes satisfy the condition* $\sum_{i \in N} b(i) = 0$ *and the minimum cost flow problem has a feasible solution.*

   We can determine whether the minimum cost flow problem has a feasible solution by solving a maximum flow problem as follows. Introduce a source node s* and sink node t*. For each node i with $b(i) > 0$, add a "source" arc $(s*, i)$ with capacity $b(i)$, and for each node i with $b(i) < 0$, add a "sink" arc $(i, t*)$ with capacity $-b(i)$. Now solve a maximum flow problem from s* to t*. If the maximum saturates all the source arcs, the minimum cost flow problem is feasible; otherwise, it is infeasible.

4. *We assume that the network* G *contains an uncapacitated directed path (i.e., each arc in the path has infinite capacity) between every pair of nodes.*

   We impose this condition, if necessary, by adding *artificial* arcs $(1, j)$ and $(j, 1)$ for each $j \in N$ and assigning a large cost and infinite capacity to each of these arcs. No such arc would appear in a minimum cost solution unless the problem contains no feasible solution without artificial arcs.

5. *All arc costs are nonnegative.*

   This assumption imposes no loss of generality since the arc reversal transformation [4] converts a minimum cost flow problem with negative arc lengths to those with nonnegative arc lengths. This transformation, however, requires that all arcs have finite capacities. When some arcs are uncapacitated, we assume that the network contains no directed negative cost cycle of infinite capacity. If the network contains any such cycles, the optimal value of the minimum cost flow problem is unbounded; moreover, we can detect such a situation by using the search algorithm [4]. In the absence of a negative cycle with infinite capacity, we can make each uncapacitated arc capacitated by setting its capacity equal to B, where B is the sum of all arc capacities and the supplies of all supply nodes.

Residual Networks

Our algorithms rely on the concept of residual networks. The residual network $G(x)$ corresponding to a flow x is defined as follows. We replace each arc $(i, j) \in A$ by two arcs (i, j) and (j, i). the arc (i, j) has cost c_{ij} and *residual capacity* $r_{ij} = u_{ij} - x_{ij}$, and the arc (j, i) has cost $c_{ji} = -c_{ij}$ and residual capacity $r_{ji} = x_{ij}$. The residual network consists *only* of arcs with positive residual capacity.

2.3.2 The Multicommodity Flow Problem

The minimum cost flow problem described in Section 2.3.1 is composed of a single commodity that must be sent from a source to a destination in some optimal fashion. This model can be extended to allow any number of commodities, each subject to individual constraints. If the commodities do not interact in any way, then we can simply solve the single commodity problem several times to arrive at a solution. If, however, the commodities interact by sharing common arc capacities, a new model must be introduced in order to find an optimal flow. This is known as the *multicommodity flow problem.*

Let x_{ij}^k denote the flow of commodity k on arc (i, j), and let x^k and c^k denote

the flow vector and per unit cost vector for commodity k. Using this notation we can formulate the multicommodity flow problem as follows:

$$\text{Minimize} \qquad \sum_{1 \le k \le K} c^k x^k \tag{2.4}$$

subject to

$$\sum_{1 \le k \le K} x_{ij}^k \le u_{ij} \qquad \text{for all } (i,j) \in A, \tag{2.5}$$

$$Nx^k = b^k \qquad \text{for } k = 1, 2, ...K, \tag{2.6}$$

$$0 \le x_{ij}^k \le u_{ij}^k \qquad \text{for all } (i,j) \in A \text{ and all } k = 1, 2, ...K. \tag{2.7}$$

This formulation has a collection of K ordinary mass balance constraints, modeling the flow of each commodity $k = 1, 2, ...K$. The "bundle" constraints tie together the commodities by restricting the total flow $\sum_{l \le k \le K} x_{ij}^k$ of all the commodities on each arc (i,j) to at most u_{ij}. Note that we also impose individual flow bounds u_{ij}^k on the flow of commodity k on arc (i,j). Many applications do not impose these bounds, so for these applications we set each bound to $+\infty$.

The MCNF problem is subject to the following assumptions:

1. **Homogeneous goods assumption.** We are assuming that every unit flow of each commodity uses 1 unit of capacity of each arc. A more general model would permit the unit flow of each commodity k to consume a given amount p_{ij}^k of the capacity (or some other resource) associated with each arc (i,j), and replace the bundle constraint with a more general resource availability constraint $\sum_{1 \le k \le K} p_{ij}^k x_{ij}^k \le u_{ij}$.

2. **No congestion assumption.** We are assuming that we have a hard (i.e., fixed) capacity on each arc and that the cost on each arc is linear in the flow on that arc. In some applications encountered in communications, transportation, and other problem domains, the commodities interact in a more complicated fashion in the sense that as the flow of

any commodity increases on an arc, we incur an increasing and nonlinear cost on that arc.

3. **Invisible goods assumption.** The model assumes that the flow variables can be fractional. In some applications encountered in practice, this assumption is appropriate; in other contexts, however, the variables must be integer valued. In these instances the model that we are considering might still prove to be useful, since the linear programming model might either be a good approximation of the integer programming model, or we can use the linear programming model as a linear programming relaxation of the integer program and embed it within branch-and-bound or some other type of enumeration approach.

2.4 The Extended Model

The MCNF problems have several things in common. A network is defined by a set of nodes and a set of arcs joining the nodes. Each arc has an associated capacity governing the amount of data that can be sent from one node to another. A commodity is a unit of information that must be sent from a source node to a destination node. Given a network and a set of commodities, the goal is to route all of the commodities through the network without exceeding the capacity on any single arc. Many routing and scheduling problems can be described as MCNF models [13].

Described briefly in Section 2.2, we have extended the MCNF problem by defining time epochs during which events can occur. Messages must be sent and arrive during predefined epochs.

In this paper we modify the standard MCNF model by turning our focus to feasibility instead of optimality. In other words, we are not concerned with finding the optimal solution, rather we are interested in whether a solution exists or not. This provides the basis for decision support. By knowing the existence of a solution, we arm a planning team with powerful information. We can show that the current resources available are indeed adequate for the upcoming mission, or that there is an excess in bandwidth that could be allocated elsewhere for increased information exchange efficiency. On the other hand, we can show that the current resources are inadequate for the given

requirements and additional bandwidth should be allocated. In addition, we can provide a planning team with exact points of weakness and possible failure points in the network. The team can then either increase bandwidth at those points or modify the communication requirements. Armed with this kind of information, Navy decision makers are empowered by added capability in decision making.

Chapter 3

Problem Statement

The application domain, Navy tactical communications networks (TCN), imposes the need for additional terminology to be related to that which is required by our mathematical models and the modeling approach.

3.1 Relevant Terminology

A communication network is defined by a set of nodes and channels. A *node* is an entity capable of performing one of the tasks of sending, relaying, or receiving messages. A node is is defined by a role-platform pair. A *role* is introduced by the TCN, and is a duty, responsibility, or function that is designated to be performed by a node in a TCN. A *platform* is an entity or location assigned to a node in a TCN. Multiple nodes can reside at the same physical location, each performing different functions. On the other hand, nodes can all perform similar functions and be located in very different geographic regions. Each unique role-platform pair defines a unique node in a communication network. In addition to roles and platforms, each node must be assigned to a functional set that includes sending, relaying, and receiving messages.

A *channel* is a communication link connecting two nodes with the following attributes: source node, destination node, bandwidth, capacity, and threshold. Bandwidth is the maximum number of bits that can be transmitted

across a channel in one second. Recognizing that an *epoch* is a time interval defined for a given mission, then each channel has a capacity or a maximum amount of data it can support in a single epoch. A threshold is assigned as a fractional value representing the proportion of maximum capacity preferred for use during a single epoch. The specification indicates a "comfortable" level of channel utilization.

A set of network links chosen for a message to define a connected path from its source node to its destination node defines a *transmission path* in a communication network. A transmission path may not contain any cycles. All potential transmission paths are known at the start of each epoch.

A *message* is a unit of information that must be sent from one node to another during a particular mission. Messages are defined by six characteristics: sending node, destination node, initiating epoch, identification number, size, and due epoch. In addition to these characteristics, the following conditions must also hold for a message to be valid: the sending and destination nodes must be defined as nodes within the communication network, and the initiating epoch must be less than the due epoch.

The resource allocation problem is centered around routing messages to their destination by a particular due date. To facilitate this, we have divided a mission into a finite number of *epochs*. An epoch is the smallest measurable unit of time that the model can recognize. An event can be the sending or receiving of a message that occurs during a designated epoch. All events take place at the resolution of a single epoch. The earliest possible epoch that a message can begin transmission is called the *initiating epoch*. A message need not be sent during the initiating epoch, but it must arrive during or before the due epoch. The *due epoch* is the latest epoch by which a message must be received. In order for a message to be received by some epoch n, it must have been sent during or before epoch $n - 1$. In addition to finding a path for a message that does not exceed any channel capacity in its transmission path, we must consider the amount of time a message takes to go from source to destination. Instead of calculating the time as the message is being routed, the model calculates a bound on the minimum size of the epoch based on the size of the largest message and the least capacity channel. The model calculates the time to send the largest message over the least capacity path in order to establish this bound. This ensures that each message can complete its transmission path in a single epoch.

3.2 Problem Statement

The goal of this research is to develop algorithms to solve the extended MCNF model described in Section 2.4 in order to provide naval decision makers with enhanced capabilities. Specifically, given a network configuration and message requirements, determine if the proposed mission is feasible in terms of the communications resources provided. Input is provided by the user in a file which defines the network and communication requirements. The program should then evaluate the mission defined in the input file to determine if the mission is feasible. The output and failure points, if any, should be displayed to the user in a clearly understandable format.

Each mission evaluation operates under one of the following two environment models:

- *The Static Model:* This model encompasses all a priori knowledge about network resources and communication requirements. As the name suggests, all input and events are known when evaluation begins and can not be changed until execution ends. This model represents a premission planning environment where changes in mission requirements are not required.
- *The Dynamic Model:* The static model extended to provide decision support during mission execution by responding to changes in assets and/or tasking. This allows the user to evaluate the effects of unsuspected changes in mission requirements while the mission is being evaluated.

The algorithms presented in this research all assume the dynamic model during execution. By operating under this model we leave it to the user to decided if changes in mission requirements are desired.

Chapter 4

Problem Approach

The NTN:RADS problem extends the multicommodity flow problem to include a time dimension divided into constant length intervals designated as epochs. The scheduling decisions available in a given epoch are dependent on decisions made in previous epochs. Therefore, independent epoch evaluation is insufficient when solving for feasibility. We have designed a set of algorithms specifically engineered to accommodate this additional complexity. This chapter presents our implementation by dissecting the program structure and examining the internal components.

4.1 Execution Environment

In the interest of portability and a strong belief in the power of object oriented programming, we chose to implement our solution in Java[1]. This allows us to execute our program on any platform that provides a Java Virtual Machine (JVM). Development occurs under GNU/Linux using GCJ [2], a Java compiler included with the GNU Compiler Collection (GCC) [1]. The core of our implementation is console-based with no graphical interface. The program is invoked using a command followed by a set of options. The command itself is the name of the executable version of the program. The command line options described below define the program execution characteristics available

[1] Java is a registered trademark of Sun Microsystems, Inc.

to determine the issue of feasibility.

- **Input file**: The name of the input file that contains the projected communications in the form of MIERs. Exactly one input file is required. All file names can be optionally preceded by a relative or absolute path name if the file is not in the current working directory. The input file format is covered in Section 4.3.
- **Output file**: The name of the output file to which message routing information is recorded. If omitted, no output file is generated. The final solution is always displayed on the console, but the execution trace is not written to a file. This option is useful for examining algorithm behavior, as it provides a record of every decision made during execution. Post processing of this file can be employed to gain further insight into topological characteristics and message interaction. The output file format is covered in Section 4.4.
- **Algorithm selection**: Selection of one of the available algorithms to be used for evaluation. Each algorithm behaves differently, allowing the user to decide which is most appropriate for a given application. The available algorithms are presented in Chapter 5.
- **Continuous vs. Incremental**: By default, the program operates in a continuous mode, evaluating each epoch in succession and presenting mission results to the user once execution is completed. The user can request that the program operate in incremental mode, where the program pauses after each epoch evaluation for inspection. During the pause the user can examine evaluation progress and make modifications to the mission requirements as seen fit.
- **Heuristics**: Enable additional algorithm heuristics that can improve run-time performance. These are execution heuristics applicable to any of the available algorithms. In certain situations, significant run time reductions are achievable. The algorithm heuristics are covered in Section 5.4.
- **Observing channel threshold**: All algorithms have the ability to ignore channel threshold parameters during execution. If channel threshold is upheld, the available capacity for any channel becomes the portion of capacity that lies within the threshold. For example, a channel's

total capacity is 10Mb and threshold is set at 80%. Upholding threshold, the available channel capacity for this channel is 8Mb. The overall effect is that the algorithm only uses a "comfortable" level of channel capacity when making message assignments. The additional capacity remains unused throughout the duration of the mission.

From the available options, the user is required to specify exactly one algorithm and the name of the input file. The remaining options are invoked depending on application.

The execution command is parsed before any data structures are populated. If the command is malformed, the program exits and notifies the user. The program tries to catch as many input errors as possible prior to execution. Any remaining errors are, hopefully, caught during run time.

4.2 Program Behavior

An illustration of the program structure is shown in Figure 4.1. After obtaining a valid set of run options, the input file is parsed and the data structures are populated. Input structure and sanity checks are discussed in Section 4.3. Evaluation begins with the first epoch after the input has been validated. As each epoch is evaluated, the scheduled message routes are recorded to the designated output file. Output record format is covered in Section 4.4. A solution is reached once all epochs have been evaluated. The solution is one of "*feasible within threshold*", "*feasible within capacity*", or "*infeasible*". The solution is displayed to the user on the console along with total execution time. A feasible solution is one in which all message successfully arrive on or before their respective due dates and no capacity on a single channel is exceeded. Feasible within threshold denotes a feasible solution where all channel utilization remains within threshold, or comfortable utilization level, throughout the entire mission. An infeasible solution indicates that no routing schedule can be found such that all messages arrive on time. The exact meaning of an infeasible solution must be interpreted within the context of the algorithm that is selected for the evaluation. The details are discussed in Chapter 5.

To improve usability, a graphical user interface is provided through which

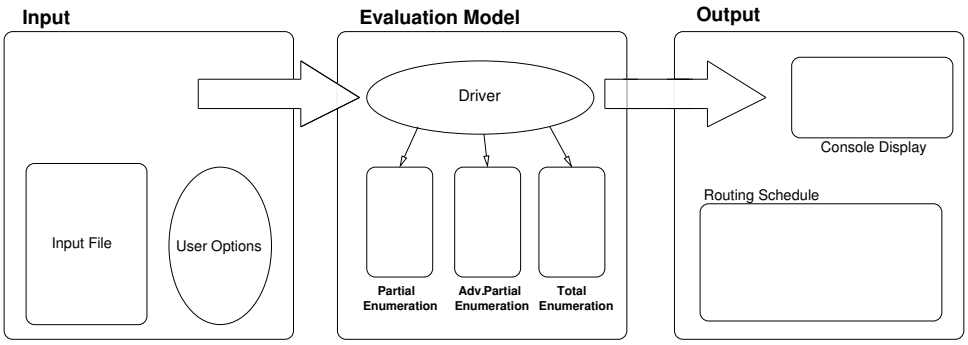

Figure 4.1: *A conceptual diagram of the program structure*

the user can manipulate the program. Algorithm selection and run options are made available through drop-down menus. Nodes, channels, and messages are displayed in separate tabs in the center of the window. Evaluation progress, including the final solution, is displayed in a tool bar along the bottom. Channel utilization and message status is updated during execution providing the same monitoring capability found in the console version. The interface eliminates the need for console interaction, improves user comprehension, and adds to the overall experience. However, the primary research objectives pertain to the algorithms and the comparative evaluation, not the interface. A robust user interface remains a possibility for future work.

4.3 Program Input

Projected communications for a mission are defined within a plain text input file. This file contains all required definitions of network topology and the communication requirements identified for a mission. An input file is composed of the following elements:

- **Problem Definition**: Defined by the following format:

  Mission Name *Number of Epochs* *Epoch Size*

  The problem definition contains the name of the mission, the number of epochs, and the epoch size. The name is a string of characters with no

embedded blanks. Epoch size is a positive real number. The number of epochs is an integer value and must be a multiple of epoch size. Exactly one problem definition can be provided in an input file.

- **Roles and Platforms**: Defined by the following format:

  ID *Name*

  Both roles and platforms are specified with a unique identification number and a name. A node may use these unique identifiers to refer to a specific role or platform during execution.

- **Node**: Defined using the following format:

  ID *Role* *Platform* *Type* *Start* *End*

  ID is a unique identification number used by messages to identify source and destination. *Role* and *Platform* are the role-platform pair that define this node. Although multiple nodes can perform the same role and multiple nodes can be located on the same platform, this node must be defined by a unique role-platform pair. *Type* is one of send, receive, or relay. A node can only perform one of these tasks during a mission. *Start* and *End* are the initiating and ending epochs for which this node is available, respectively.

- **Channel**: Defined using the following format:

  ID From To Type Bandwidth Threshold Cost Begin End

  The *ID* is a unique number that identifies the channel. *From* and *To* are the two nodes connected by this channel. *Type* is a name that designates channel type. For instance, two nodes can be connected by two channels where one channel only supports radio transmission and the other supports 802.11 wireless communication. *Bandwidth* is defined in terms of megabits per second. *Threshold* is a percentage of total capacity and thus must be a number between 0 and 100. *Cost* is a placeholder for extension of the resource allocation decision problem to more than the feasibility issue. Currently our algorithms do not consider this attribute during execution, leaving this a topic for future investigation. *Begin* and *End* are epoch values for which this

channel becomes available and unavailable for message transmission, respectively.

- **Message**: Defined using the following format:

  From *To* *Start* *ID* *Size* *Due* *Type*

  From and *To* are the sending and receiving nodes, respectively. *Start* is the initiating epoch in which the algorithm can begin considering this message for scheduling. *ID* is a unique identification number used to distinguish between messages that have the same source, destination, start epoch, size, and due epoch. *Size* is the length of the message in Megabits. *Due* is the epoch by which this message must arrive. In order for a message to arrive in epoch n, it must be sent at the latest, in epoch $n - 1$.

Before execution begins, the program verifies that the input file contains valid definitions for all of the aforementioned elements. A well-formed input file exhibits the following characteristics:

- A single problem definition must be specified.
- At least two nodes, one channel, and one message must be defined.
- All references to roles, platforms, and nodes by messages and channels must be valid. The program verifies each reference for correctness.
- All epoch references must be non-negative, and within the epoch range defined in the problem definition.
- All message sizes and channel bandwidths should be non-negative and any message size should be less than at least one channel capacity in the network.

Once all checks are completed, data structures are populated and the algorithm begins execution with the first epoch.

Each node and channel has a beginning and end epoch, allowing a user to define network topologies and capabilities that change dynamically throughout the course of mission evaluation. Changes in communication requirements or

capabilities cannot occur during evaluation. Any changes in network topology or capabilities must be present in the input file prior to execution. This feature provides the ability to simulate some interesting scenarios. For instance, a node can be defined that communicates only with neighboring nodes as it changes position in the network over time. A change in channel capacity is represented by removing a channel from the network during an epoch and replacing it with a channel of higher or lower capacity.

4.4 Program Output

Output is generated in two places, the console and the output file. Console output consists of the final solution, total execution time and any other requested information regarding mission evaluation. This output is displayed on the console where the program is executed.

File output is a comprehensive record of every routing decision made by the algorithm during execution. This output is written to an output file, the name of which is specified as a command line argument to the program. Incidentally, if no output file name is defined then no output file is created. There are only three scheduling events that can take place during execution: *message sent*, *message postponed*, and *message failed*. Sent and failed are self explanatory. A message is postponed in an epoch when it is available to be sent but transmission is not required to satisfy its due date. Further discussion of these events is available in Chapter 5. The format of each event record is described below.

- **Send**: Defined by the following format:

  S *Epoch* *Index* *ID* *Size* *Path*

  The send record is identified by the character *S* at the start of the line. Each time a message is successfully sent from source to destination, a send record is generated of this form. *Epoch* is the epoch in which the message is sent. Because messages can be resent over different paths and in different epochs, *Index* is used to distinguish this event from others having identical attributes otherwise. *ID* is the message identification string as described in Section 4.3. *Size*, although already

contained in the message ID, is provided again to improve readability. *Path* is the actual path assigned to a message during transmission. A path has the following format:

Length N_S *ChID,RC* N_1 ... N_D

Length is the total number of hops in the path. Following the length is a variable length path list that starts with the source node and ends with the destination node. Between every node pair is a *ChID,RC* element that identifies the channel used to travel from the node on the left to the node on the right. *ChID* is the channel identification number. *RC* is the remaining capacity on that channel after the message has completed transmission.

- **Postpone**: Defined by the following format:

  P *Epoch* *ID* *Size* *Source* *Dest*

  The postpone record is identified by the character *P* at the start of the line. *Epoch* is the epoch in which the postponement occurred. This is followed by the postponed message's attributes starting with the message identification string, *ID*. *Size* is the size of the message. *Source* and *Dest* are the source and destination nodes of the postponed message, respectively.

- **Failure**: Defined by the following format:

  F *Epoch* *Index* *ID* *Size* *Source* *Dest* *Network*

  The failure record is identified by the character *F* at the start of the line. *Epoch* is the epoch in which the failure occurred. *Index* is the unique identification number for this failure. Messages can fail at the same point in time any number of times. The index number is used to distinguish between otherwise identical failures. *ID* is the message identification string for the failed message. *Size* is the size of the message. *Source* and *Dest* are the source and destination nodes of the failed message, respectively. Following the destination node is a list of channels and associated residual capacities. This list represents the current state of the network. When a message fails, in addition to knowing which message causes the failure, we would also like to see a snapshot

of the entire network at the time of failure. This can be used not only for debugging purposes, but also to provide information to the user about the state of the network when a failure occurs. At this point we can examine the current state of the network and possibly identify problem areas that exist.

Possessing the above information derived from execution, the user hopefully can answer virtually any question regarding mission feasibility evaluation.

4.5 Summary

To accommodate the additional requirements of the NTN:RADS problem, the standard MCNF problem is extended to incorporate a time element. A mission is divided into constant length time intervals called epochs. The epoch allows a user to identify specific points in time during the course of a mission, such as message arrival requirements and node availability specifications. A solution to this problem is provided by three separate algorithms written in Java and compiled using GCJ. Input validation is the first action taken during program execution. Upon receiving a valid input, the user-selected algorithm evaluates the mission to obtain a solution. A solution can be any one of: "feasible within channel threshold", "feasible within channel capacity", or "infeasible." In addition to the feasibility assessment, the program records each scheduling assignment made during execution, providing a user with all the necessary information to make a decision regarding communication feasibility.

Chapter 5

Algorithm Design

Three algorithms developed for the NTN:RADS models are defined in Section 2.4. This chapter provides a detailed description of each algorithm including a comparative analysis in addition to expected results. To further illustrate the differences among algorithms, a simplified example is presented that documents the logical steps required in applying the algorithms to determine a solution.

5.1 Algorithm Approach

The combinatorial nature of the NTN:RADS problem, and MCNF problems in general, has a significant impact on available solution strategies and algorithm design options. Before detailing the elements of our design, some common terms used for description and analysis are defined as follows:

- **Input Space**: An *input* is a set of parameter values, where each message is defined by value assignment to the parameters identified in Chapter 4. The same is true for the definition of nodes and channels and the value assignments to their parameters. An *input space* is an abstraction used to represent the set of all possible input values. Any modification to a file containing input information is referred to as a change of input values within the input space.

- **Routing Schedule**: A set of message assignments, where each assignment is defined by a message identifier, the epoch during which the message was sent, and the path taken by the message from source to destination. A *complete* routing schedule contains a message assignment for every message defined in the corresponding input space. All routing schedule references in this paper are assumed complete unless stated otherwise. A routing schedule defines one solution for a particular input space. Therefore, the primary goal of any algorithm lies in finding a feasible routing schedule as quickly as possible.
- **Solution Space**: A *solution space* is the set of all possible routing schedules that can be generated from a given input space.
- **Feasible Solution Space**: Considering all the solutions contained in a solution space, only a subset provides a feasible outcome. The set of feasible solutions within a solution space is known as the *feasible solution space*. This space can contains as few as zero solutions, or as many as the entire solution space itself. Exact dimensions are purely dependent on the multi-variable interactions among the parameters of an input space.
- **Search Space**: Each of the three algorithms discussed in this chapter defines its own *search space*, a possibly improper subset of a solution space. An algorithm's search space is the set of all routing schedules that an algorithm can generate during execution on a given input space. The number of elements in this space is determined by the number of parameters considered in producing potential solutions. Algorithms searching only a subset of the parameters produce combinations representing only a subset of the total solution space. In contrast with an algorithm that evaluates all parameter combinations (total enumeration), such algorithms attempt to find a feasible solution more quickly. This concept is discussed further in following sections.

Due to the complex interactions among the large number of parameters that define the input space, one cannot make general conclusions about algorithm efficiency or effectiveness. Both of these measures are dependent on the size and location of the search space and feasible solution space with respect to the total solution space. Predicting any of these values prior to algorithm execution is impossible.

The guarantee that no feasible solution exists for a given input (set of parameter values) can be achieved only by considering all possible routing schedules. This is equivalent to defining a search space equal to the solution space. For many input instances, the time required to generate every possible routing schedule is prohibitive. To help alleviate some of the computational overhead, a tiered execution model forms the basis of our approach to solving this problem.

An obvious simple solution is to evaluate very possible routing schedule searching for a feasible solution. Ultimately, this type of approach might be unavoidable. However, the approach in this research uses heuristics in the attempt to determine the existence of a feasible solution much quicker than would be realized by total enumeration.

This chapter presents the three algorithms that define the tiered execution model designed to provide feasibility assessment for tactical networks. Each algorithm leverages a different set of decision-making capabilities during the course of execution, allowing the user to weigh the importance of efficiency and effectiveness when selecting an algorithm.

5.2 Common Algorithm Components

Each algorithm differs in the extent to which it is permitted to make scheduling decisions. Slight differences in scheduling decisions might lead to significant differences in search space size. Each algorithm is built on an identical execution nucleus and uses the same data management scheme during execution. This section describes these common components by detailing the structure of the execution model and data management scheme forming the nucleus of all three algorithms.

5.2.1 The Execution Model

The benefits of recursion became apparent during the design phase of our solution. Implementation wise, a stack is very useful. Recursion provides implicit construction and maintenance of the stack, allowing an algorithm to easily keep track of current and previous states. A state is simply a partially

constructed routing schedule in which one or more messages have path assignments. At any time, an algorithm can retrace previous decisions to try an alternative path for some message. Retracing steps to allow alternative path components to be evaluated is called "backtracking." The number of conditions under which backtracking is permitted establishes the fundamental differences among the three algorithms.

A closer examination of recursive behavior reveals three distinct components that are subject to re-evaluation: *epoch*, *message*, and *path*. All three are interrelated and exhibit a great deal of interaction as shown in Figure 5.1.

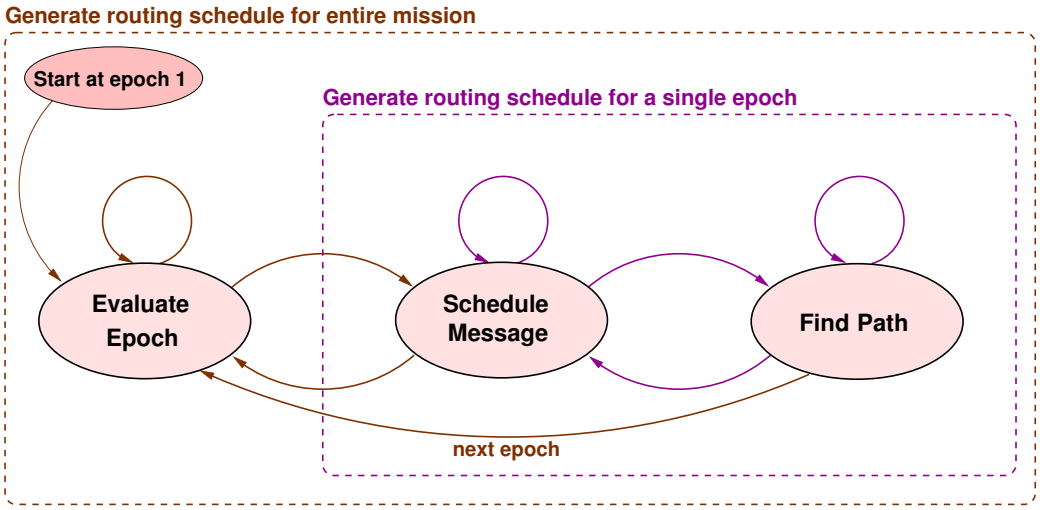

Figure 5.1: *Illustration of the Recursive Framework*

Evaluation begins with the first epoch. At the highest level is epoch recursion. Upon successful evaluation of a single epoch, the epoch recursive procedure is then called again for the next epoch. The return value of an epoch recursive procedure dictates the course of action that the previous epoch evaluation should take. If a routing schedule cannot be found for a given epoch, the previous epoch recursive procedure has the ability to modify the routing schedule and reevaluate the next epoch. If no modification to the routing schedule is possible, the evaluating procedure can declare failure itself and allow the prior epoch evaluation to try an alternative. This process continues until every epoch is successfully evaluated and a feasible routing schedule is obtained or no routing alternatives allow any further progress. The ability to

step back to previous epochs and make modifications is dependent on which algorithm is used for evaluation.

Little logic is contained in epoch recursion. All of the work required to evaluate an epoch is handed off to message recursion. Within an epoch, messages are assigned to paths in the network. The message recursive procedure is responsible for finding a path assignment for a single message. Similar to the epoch recursive process, the message recursive procedure is called for each message until the network reaches its capacity. Using message backtracking, a message can be reassigned to a different path in order to allow other messages to reach their destination during an epoch. Like epoch backtracking, the ability to backtrack on messages is algorithm dependent.

Instead of making path assignments, the message recursive procedure passes that responsibility to the path recursive procedure. Path recursion involves the identification of a set of links through the network to connect the source and destination nodes. Path construction follows a simple set of rules. A path starts at the source node for a message. The first step is to look for a direct path from the source to the destination. If one exists, the direct link defines the path. If a direct link is unavailable or an indirect path is required, the search extends to the first untried relay node with sufficient residual capacity. The relay node link is added to the path, and becomes the new source. From here the processes repeats. If no direct link or relay node can be found, the current link is deleted from the path and the search is re-initiated at the previous node. No path from the source denotes path failure, which is reported to the procedure that invokes path recursion for this message. If a successful path is discovered then the path recursive procedure calls message recursion for the next message. These two procedures alternate in passing control until all messages for a given epoch are scheduled. The path recursive procedure then calls epoch recursion to evaluate the next epoch.

The recursive procedures described above work together during evaluation to find paths through the network, assign paths to messages in an epoch, and evaluate all epochs defined in the input space. This framework governs the pursuit of a feasible routing schedule. However, each algorithm has the ability to backtrack only at certain points, making each algorithm behave differently.

5.2.2 Data Management

The data management scheme is comprised of two components: storage and selection. Storage refers to how the parameters in the input are arranged and maintained. Selection refers to the specific input parameters needed by the epoch evaluation procedure for a particular epoch. The details follow.

Storage

Nodes, channels, and messages comprise sets that are stored in distinct, linked lists. In the case of nodes and channels, the list has no particular ordering. Location is solely dependent on the order defined in the input file. The message list, however, is sorted in the order of earliest due date. The importance of this ordering becomes evident in the following sections.

In the case of partial and advanced partial enumeration, where the search space is a proper subset of the solution space, messages required to arrive earlier must be scheduled prior to messages that can arrive at a later date. By maintaining a sorted message set, the algorithm is forced to this behavior. Such behavior gives both algorithms the best possible chance of finding a feasible routing schedule with little additional backtracking. For example, if an undesirable decision is made in a prior epoch, neither algorithm is permitted to backtrack across an epoch boundary in search of an alternative.

Total enumeration differs from the other two algorithms in that it defines a search space equal to the solution space. Thus, the final solution, feasible or infeasible, is always correct. The ordering of the message set has no effect on the final solution because all possible combinations are eventually considered. However, due date ordering does keep total execution time to a minimum. In no case for these three algorithms would a violation of due date ordering prove beneficial.

Selection

Since messages have a window during which they can be sent, consideration of the entire message set in every epoch is unnecessary. Likewise, nodes and channels are only available during a certain window of time. Considera-

tion of every node and channel during a single epoch is equally unnecessary. Therefore, a subset of network components and messages is selected prior to each epoch evaluation. Subset selection allows the epoch evaluation procedure of an algorithm to focus only on relevant information during message scheduling, thus reducing the time required to do so.

The message set for each epoch is constructed simply of the available messages that have yet to be scheduled. This set remains sorted in the order of earliest due date and is passed along with the custom epoch-specific network components to the corresponding epoch evaluation procedure for that epoch.

A custom communication network is constructed for each epoch based on the nodes and channels available in that epoch. Thus, a user can provide a network specification that changes from one epoch to another. For a node to be included in the network for a specific epoch, it must be available during that epoch and be connected by at least one available channel to some other available node. If a channel is available in an epoch but its two connecting nodes are not available, the channel is ineligible for inclusion in the epoch. This approach provides an algorithm with the smallest possible set of data on which to operate for each epoch, thus reducing total execution time.

When a node or channel is considered *available* during a given epoch, it is assumed to come into existence some time during the previous epoch. All events take place at the granularity of a single epoch. Therefore, message transmission requires all components of a transmission path to be available for the duration of the epoch.

5.3 Specific Algorithm Behavior

Each algorithm tries to schedule as many messages as possible in each epoch. Message postponement is only considered where no other option is available. This type of greedy scheduling is called "work conserving scheduling." By employing a work-conserving model in each epoch, we hope to find a feasible solution as quickly as possible. Initially, each epoch is evaluated with the knowledge that the maximum number of message assignments has occurred in the previous epoch. However, this technique may not always produce the best global result. Therefore, in order to guarantee correctness of the

feasibility solution, all possible combinations of messages in each epoch must be considered.

What constitutes a feasible solution? Successful scheduling of the last message in the last epoch implies that all prior messages are also successfully scheduled. Forward progress is made only after a successful scheduling decision. In the search for a feasible solution, an algorithm might use backtracking to reassign messages any number of times during execution. A mission is feasible if, and only if, the last message in the last epoch is successfully scheduled. An infeasible solution implies that no message reassignment exists to allow forward progress. The rules for allowable message reassignment differ among the algorithms and these differences are the subject of the following section.

5.3.1 Partial Enumeration Algorithm

The Partial Enumeration Algorithm (PEA) is the simplest of the three algorithms. The PEA uses a recursive, backtracking procedure only to find paths for messages. It neither reassigns messages within an epoch nor reevaluates prior epochs. At no time is message reassignment permitted to occur. Any message failing to arrive by its due date results in an infeasible solution.

Because only a single permanent assignment is made for each message, this algorithm is able to evaluate its search space much quicker than the other two algorithms, where finding a feasible solution requires message and/or epoch backtracking. This is especially the case as the input space becomes large. The obvious drawback to this approach is the inability to use the other forms of backtracking to find a feasible solution. Although a feasible solution found by this algorithm is *always correct*, the determination of an infeasible solution may not be correct. This case may require the use of one of the more advanced algorithms to find a feasible solution.

Although an infeasible solution is ultimately inconclusive, it may in fact provide insight into some of the characteristics of the particular problem that defines the input space. In particular, an infeasible solution may indicate an imbalanced communication network. It may also indicate that communication requirements closely match the available resources. This may be undesirable if the user is looking to maintain a large percentage of unused re-

sources for emergency purposes during the course of an operation. Depending on application, these inferences could prove to be useful during a planning phase without requiring the use of one of the more advanced algorithms.

5.3.2 Advanced Partial Enumeration Algorithm

The Advanced Partial Enumeration Algorithm (APEA) takes PEA a step further to provide feasibility assessment. APEA is permitted to make any number of message reassignments within any single epoch. It may not, however, reschedule previously evaluated epochs. Message backtracking allows this algorithm to consider a much larger number of scheduling combinations and, thus, increases the size of the search space by an exponential factor. This added scheduling flexibility provides an advantage over PEA by increasing the probability of finding a feasible solution. The added flexibility does come at a cost, however. Test results presented in Chapter 6 show the potential for dramatic increases in run time requirements over that of PEA. The hope is that by relaxing message reassignment constraints, a feasible solution unrecognizable by PEA can be found in a reasonable amount of time. It is left to the user to decide if this trade off is acceptable.

5.3.3 Total Enumeration Algorithm

The Total Enumeration Algorithm (TEA), as the name implies, defines a search space equal to the total solution space. That is, TEA considers every possible scheduling combination during the course of execution. Total enumeration is achieved by extending the capabilities of APEA to allow epoch backtracking. At this level, any message reassignment in any epoch is permitted. The resulting solution is conclusive. The possibility of a feasible solution exists when PEA or APEA report infeasibility. Thus, TEA stands alone as the only algorithm of the three to always provide a conclusive answer. However, like APEA, the exponential increase in search space size can result in similar increases in execution time requirements. Results illustrating this fact are presented in Chapter 6.

5.3.4 Efficiency vs. Effectiveness

The above sections describe three algorithms, each having trade-offs in terms of efficiency and effectiveness. On one end of the spectrum, PEA provides maximum efficiency at the cost of providing a possibly incorrect feasibility assessment. APEA falls in the middle, with a better balance between efficiency and effectiveness than the other two algorithms. TEA lies at the opposite end, providing ultimate effectiveness with the potential for unacceptable execution time requirements. The importance of these two factors can only be weighed by the user. The intent of providing these options is to achieve a level of flexibility previously unattained in the area of feasibility assessment for tactical networks.

5.4 Additional Heuristics

The combinatorial nature of the MCNF problem inspires the application of additional heuristics in an effort to help further reduce execution time requirements. Two heuristics employed by all algorithms are presented in previous sections: *due date ordering* and *work conserving scheduling*. Of the remaining heuristics recorded during the course of this research, one is included in our implementation. This is refereed to as *precalculating message paths at the start of each epoch.*

More than a single message is likely to have the same source and destination. As described in Section 5.2, a path is constructed for each message as it is evaluated. If multiple messages have the same source and destination, then time is taken to calculate the same path for each message. We can therefore remove some redundant computation by calculating all paths from all sources to all destinations at the start of each epoch. Rather than constructing a path from each source to destination individually, we can improve run time by simply looking up a path in the precalculated list. Results of this heuristic are detailed in Chapter 6. Other heuristics have been identified and included as potential future work in Chapter 7.

5.5 A Simplified Example

For added clarification and understanding, we offer a simple example to illustrate the differences in the algorithms. Although highly simplified, the general concepts and algorithm distinctions are present, hopefully achieving the goals of the demonstration.

A simple network is shown in Figure 5.2. Node one in this network acts as a sending node, node four is the receiving node, and nodes two and three are responsible for message relaying. Each channel has two numerical values given as labels. The label to the left is the total channel capacity. The label to the right in brackets is threshold capacity. For this example, channel threshold is set to 80% for all channels. The resulting threshold capacity value is simply 80% of the channel capacity. The purpose of channel threshold is to allow an algorithm to make message assignments within a certain utilization comfort level. If channel threshold cannot be maintained while searching for a feasible solution, the algorithm falls back to using the total channel capacity. For the purposes of this example we are going to consider only the actual capacity figures and ignore channel threshold. To further simplify this example, epoch size is set to one and the mission duration requires three epochs.

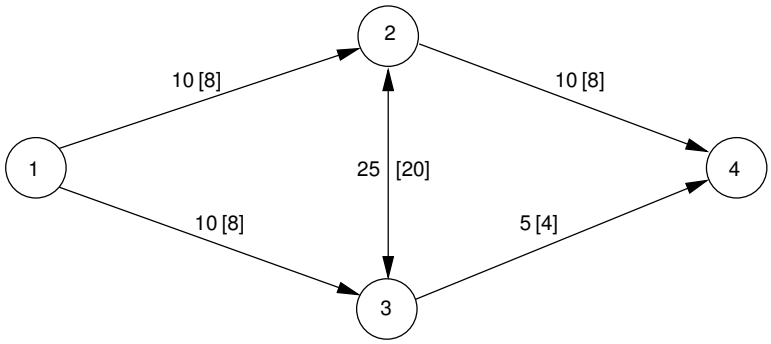

Figure 5.2: *An example network with five nodes and five channels*

The message set, shown in Table 5.1, consists of six messages, three of which must be sent in the first epoch, and the remaining three in the second. The sending window for all messages is one epoch in length, so messages one, two, and three *must* be sent in epoch one, and messages four, five, and six *must*

be sent in epoch two.

Message ID	*From Node*	*To Node*	*Size*	*Initiating Epoch*	*Due Epoch*
1	1	4	5	1	2
2	1	4	5	1	2
3	1	4	5	1	2
4	1	4	4	2	3
5	1	4	5	2	3
6	1	4	6	2	3

Table 5.1: *An example set of messages prior to sorting*

5.5.1 Partial Enumeration

Once the network and communication requirements are defined, an algorithm can begin evaluation. We are using partial enumeration for the first evaluation. Before execution begins, the message set is sorted in order of earliest due date, resulting in the sorted set shown in Table 5.2.

Message ID	*From Node*	*To Node*	*Size*	*Initiating Epoch*	*Due Epoch*
2	1	4	5	1	2
3	1	4	5	1	2
1	1	4	5	1	2
5	1	4	5	2	3
6	1	4	6	2	3
4	1	4	4	2	3

Table 5.2: *An example set of messages after due-date ordering*

Once the global message set is sorted, both the network and message set are constructed for epoch one. For this example, all nodes and channels in the network remain available throughout the entire mission. Therefore the network for epoch one is identical to that given in Figure 5.2. The epoch one message set consists of messages one, two, and three. Upon construction of the message set and network for epoch one, partial enumeration evaluates the first epoch making the following scheduling decisions:

Step	*Message ID*	*Status*	*Path*
1	2	Sent	$1 \rightarrow 2 \rightarrow 4$
2	3	Sent	$1 \rightarrow 2 \rightarrow 4$
3	1	Sent	$1 \rightarrow 3 \rightarrow 4$

Table 5.3: *Message assignments made by PEA during epoch 1*

Evaluation begins when the algorithm makes a call to the epoch recursive procedure for the first epoch. The message recursive procedure is then called for the first message in the message set. As Table 5.2 shows, message two is the first message the algorithm attempts to schedule. The path assignment for message two can be seen in row one of Table 5.3. After making a successful message assignment, the message recursive procedure is invoked for the next message in the list. Message three is assigned to the path described in row two of Table 5.3. The message recursive procedure is invoked once again for the last remaining message in this first epoch. Row three of Table 5.3 shows the message assignment for message one.

At this point in the evaluation the call stack has four elements illustrated in Figure 5.3.

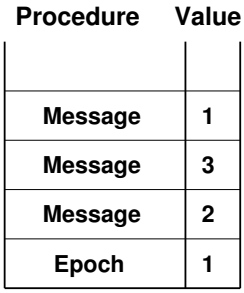

Figure 5.3: *The call stack after PEA schedules the last message in epoch 1*

The epoch recursive procedure makes a call to the first message recursive procedure which, in turn, calls itself consecutively for every available message in the message set. Once the last message in the set is reached and a successful assignment is made, each message recursive procedure is popped off the call stack and the epoch recursive procedure invokes epoch recursion for the next epoch.

On taking a closer look at the routing decisions made in Table 5.3, we see that message one is routed differently than the previous two messages. After message three is sent, the residual capacity on the channel connecting node one with node two is zero. Therefore, in order for message one to reach its destination, it must be routed through another relay node. In this case, node three is used as a relay to reach the destination. Finding this alternate path requires backtracking at the path level. Path backtracking, as described in Section 5.2, is permitted by all algorithms.

After success in epoch one, PEA now moves on to the second epoch. Table 5.4 shows the scheduling decisions made by partial enumeration during epoch two.

Step	*Message ID*	*Status*	*Path*
1	5	Sent	$1 \rightarrow 2 \rightarrow 4$
2	6	**Failed**	
	4	Unsent	

Table 5.4: *Message assignments made by PEA during epoch 2*

At step one in epoch two, message five is successfully sent from source (node one) to destination (node four) via node two. At step two, no path is available to allow message six to reach its destination during epoch two, resulting in message failure. Single message failure in partial enumeration results in mission infeasibility. The network snapshot, shown in Figure 5.4, depicts the state of the network just before the failure occurs.

Clearly, if message five had taken a different path to its destination, via node three perhaps, message six would have arrived successfully. However, partial enumeration does not permit reassignment of messages under any circumstance. Therefore, execution is terminated and the mission is determined to be infeasible.

5.5.2 Advanced Partial Enumeration

Using the same set of input parameters, we now examine how advanced partial enumeration handles the failure of message six in epoch two. The goal is to recognize the message failure and make a reassignment to allow successive

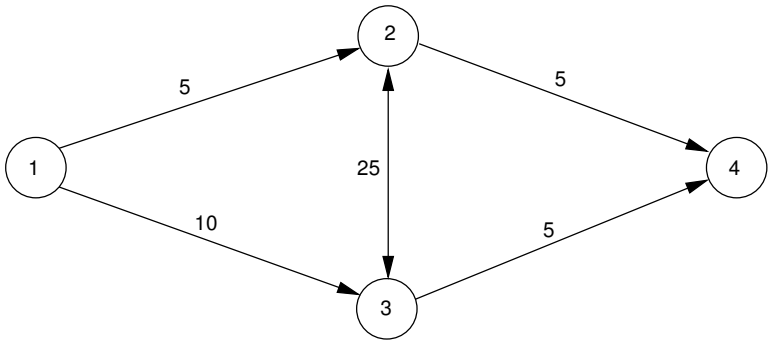

Figure 5.4: *A snapshot of the network at the time that failure occurred*

messages to be sent. In this case, message six needs to be reassigned so that message five can be sent. The end result being a feasible solution.

Because advanced partial enumeration is an extension of partial enumeration, the same set of logical steps are followed as in partial enumeration during epoch one. The difference among algorithms lies in the ability to backtrack once a failure is discovered.

After success in epoch one, Table 5.5 shows the steps taken by advanced partial enumeration during epoch two.

Step	*Message ID*	*Status*	*Path*
1	5	Sent	$1 \rightarrow 2 \rightarrow 4$
2	6	**Failed**	
3	5	Resent	$1 \rightarrow 2 \rightarrow 3 \rightarrow 4$
4	6	Sent	$1 \rightarrow 3 \rightarrow 4$
5	4	Sent	$1 \rightarrow 2 \rightarrow 4$

Table 5.5: *Message assignments made by APEA during epoch 2*

Here we see that upon failure of message six, backtracking occurs and message five is resent along a different path. This reassignment allows message six to be successfully transmitted to its destination followed by message four. Because of a single message reassignment, a feasible solution is now obtained.

5.5.3 Total Enumeration

Again, total enumeration is an extension of advanced partial enumeration. Therefore, the same set of logical steps are following in arriving at the same solution. This example, unfortunately, does not provide a situation in which the logical steps would differ. The third tier of the tiered execution model is not accessed because a feasible solution is achieved in tier two.

5.6 Summary

The MCNF problem falls into the category of NP-Complete. Therefore, we provide a tiered execution model to provide feasibility assessment in the presence of this combinatorial hurdle.

The model of our approach contains five components. An *input space* is defined as a set of parameter values contained in an input file. A *routing schedule* is a set of message assignments where every message is assigned a path in some epoch. A *solution space* is defined as the set of all possible routing schedules for a particular input space. The *feasible solution space* is the set of routing schedules within the solution space that represent feasible solutions. Each algorithm defines a *search space* that is a subset of the solution space. The search space contains the set of routing schedules that an algorithm is capable of generating from an input space. The complexity of the multi-variable interactions among the parameters within an input space makes run-time predictions difficult.

Though each algorithm is different, each shares the same nucleus. The execution model is based on a recursive design that takes advantage of a stack to keep track of state information. Data management is comprised of storage and selections. Each set of parameters is stored and maintained in a linked list and messages are sorted in the order of earliest due date. Only the available members of the input space are submitted to each epoch in an effort to reduce total execution time.

Of the three algorithms, *Partial Enumeration* makes only one assignment per message during execution. No message reassignment is allowed to occur within or across an epoch. Results from this algorithm may provide valuable

insight into the overall nature of a given input.

Advanced Partial Enumeration extends PEA to allow message reassignment only within a single epoch. This comes with the cost of increasing the size of the search space, and thus, potentially increasing execution time requirements. Because the search space is larger than that of PEA, this algorithm may have a better chance at finding a feasible solution if one exists.

Total Enumeration further extends APEA to allow full flexibility in message reassignment. Though the final solution is conclusive, execution time can be exponentially greater than that of APEA. TEA defines a basis from which to judge the effectiveness of the other two algorithms. It also provides a benchmark for looking at any further or additional potential heuristics.

Given these choices, the user is free to weigh the importance of efficiency and effectiveness when selecting an algorithm. Further, different requirements (advanced planning versus real-time decision support) can produce differing priorities assigned to efficiency and effectiveness.

Chapter 6

Analysis and Results

Chapter 5 presents the three spaces: search, solution, and feasible solution space. The dimensions of each of these spaces depends upon the numerous multi-variable interactions among attributes contained in the input space. These attributes combine to define the size of each space and the routing schedules contained within. The size of each of these spaces and the location of the feasible solution space with respect to the search space determine how an algorithm behaves during execution. In addition, these factors contribute heavily to the time required for an algorithm to determine feasibility for a given input space.

This chapter attempts to provide insight into algorithm behavior by exposing the dominant factors influencing total execution time. We seek to explore the major factors influencing algorithm behavior by providing a conceptual characterization of the problem domain based on our understanding of the problem coupled with experimental results gathered during testing. Although testing results support the conceptual claims, some interesting discoveries have become evident during testing. Additionally, we present comparisons with an algorithm augmented by path precalculation, and show that for equal effectiveness, significant gains in efficiency can be achieved.

The results and analysis presented in the chapter are based on observations made during development and testing. While we have results to support the claims made here, we cannot claim these results to be general or comprehensive.

6.1 Conceptual Characterization

The overall efficiency of an algorithm is governed by the size of the search space and the methods employed to evaluate routing schedules contained within. The details of these two primary factors are described in the sections to follow.

6.1.1 The Search Space

Considering the worst case, an algorithm evaluates every routing schedule within the search space before determining that no feasible solution exists. Therefore, worst-case execution time is relative to the size of the search space for a given algorithm and input space. An illustrative relationship among search and solution spaces is found in Figure 6.1.

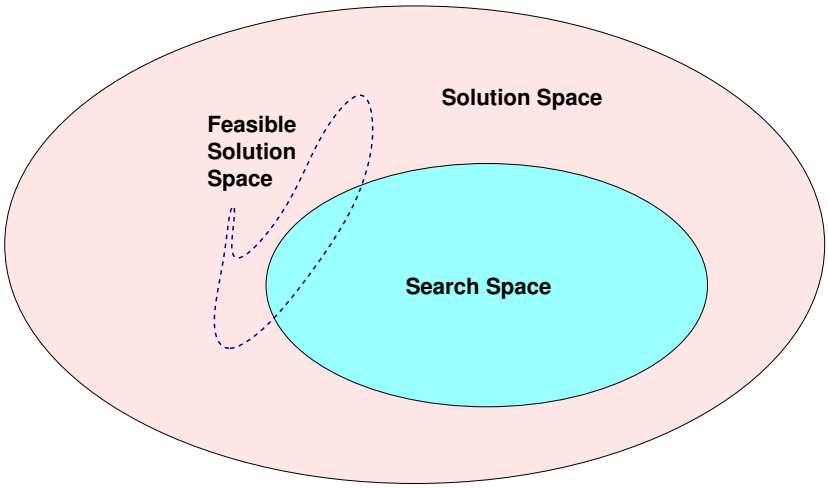

Figure 6.1: *An illustrative relationship among solution, search, and feasible solution space*

In this figure, the blue area labeled "search space" contains the set of routing schedules that an algorithm is capable of generating during execution. The pink area labeled "solution space" is the set of all possible routing schedules for a particular input space. The feasible solution space, defined by the dotted line, is the set of all routing schedules that result in a feasible solution. The

feasible solution space intersects the algorithm search space, implying that this algorithm can find a feasible solution during execution.

In a more accurate model, Figure 6.1 would depict an n-dimensional hyperspace due to the large number of interactions among input parameters. However, we have simplified this illustration to a representation in two dimensions. This simplified model proves more than sufficient explaining analysis presented in the following sections.

Each point within a search space represents a routing schedule that an algorithm can generate. Evaluation begins at some point within the search space and progresses in no well defined pattern. Feasibility is established once the algorithm generates a routing schedule that lies within the feasible solution space. The amount of time required to find such a routing schedule is difficult to predict. However, describing the most influential determining factors of search and solution space characteristics allows a user to take better advantage of data returned by an algorithm after execution completes.

- *Network topology.* The average number of paths from a source to a destination plays a key role in defining the search space dimensions for an algorithm. In general, the number of unique paths available for any single message assignment is directly proportional to the size of the search space. However, these paths must have sufficient capacity to allow message transfer. Increasing the average number of paths available for a message provides a backtracking algorithm with additional alternatives in routing assignment. This increased flexibility results in higher execution times in most cases.
- *Number of messages per epoch.* The average number of messages available for transmission during a single epoch is directly proportional to search space size. As this factor increases, the number of routing combinations an algorithm must consider increases significantly.
- *Message window size.* The average size of the time window in which a message can be sent has a large effect on scheduling flexibility. A larger window size allows an algorithm greater freedom in making message assignments. If a feasible solution exists within the search space, this factor can allow an algorithm to find a solution in less time. Likewise, if no feasible solution exists, this factor can increase execution time dramatically. Smaller window sizes tend to restrict the ability of an

algorithm to find a feasible solution while at the same time decreasing total execution time.

- *Message size.* The consequences of message size can be explained by using the classical knapsack problem as a model. Using an epoch as a knapsack, the goal is to maximally pack the sack leaving the least possible amount of waste. In general, while increasing scheduling options, smaller message sizes can increase overall execution time if the average number of message paths increases as a result. On the other hand, larger message sizes limit scheduling options and can reduce execution time. However, the reduced number of scheduling options may prevent an algorithm from finding a feasible solution.

The above mentioned factors are all tightly coupled. Additionally, other unmentioned factors could play a contributing role in determining search space size. In most cases, a change in one of these factors affects the others. This relationship makes algorithm behavior almost impossible to predict and prevents single parameter sensitivity investigation. However, awareness of the core factors that influence algorithm behavior enables some general conclusions regarding input space characteristics.

6.1.2 Heuristic Effects on Algorithm Behavior

The ability to make general predictions regarding algorithm behavior relies on the use of heuristics by each algorithm during execution. Generating routing schedules with randomly assigned components, a common strategy in experimental investigation, does not have meaning in the NTNRADS context. This section describes how heuristics affect algorithm behavior and support meaningful analysis based on intelligent scheduling logic and predictable behavior. Refer to Chapter 5 for a complete description of the available heuristics.

Each algorithm employs due-date ordering and work conserving scheduling to improve efficiency and make intelligent scheduling decisions. The payoff in using these heuristics is dependent on whether a feasible solution exists within the search space of an algorithm. In the event that no feasible solution is available, each algorithm resorts to generating every possible routing schedule within its search space before completing execution with an infeasible solution. However, in the event that a feasible solution does exist within

the search space, these two heuristics appear to have a dramatic effect on algorithm efficiency, i.e. decreasing the time required to find a feasible solution.

The use of heuristics allows each algorithm to make "good" routing decisions early in the execution. A good routing decision is one that increases the chance of finding a feasible routing schedule with minimal backtracking. As some routing choices are obviously better than others, the two heuristics are used by the algorithms to improve scheduling decisions and hopefully decrease execution time by finding a feasible solution faster. This concept can be viewed as shortening the distance between the initial point in the search space and the closest point in the feasible solution space.

Intelligent scheduling is accomplished in the following manner. By ordering messages according to their due dates, an algorithm attempts to schedule messages of higher priority before those with lower priority. In no case does it prove beneficial to schedule a message required to arrive later before a message required to arrive earlier.

Work conserving scheduling further increases intelligent scheduling by forcing an algorithm to maximally pack each epoch. Like due-date ordering, leaving uncapitalized resources in previously evaluated epochs is never desirable. Combined, these two heuristics allow an algorithm to begin an epoch evaluation with the knowledge that many "good" scheduling decisions are incorporated in prior epochs. Consequently, the likelihood is that a feasible solution can be found in less time.

6.2 Experimental Results

This section presents run-time measurements gathered during algorithm testing. Our intent here is to test each algorithm with a representative set of input values. In doing so, several conjectures emerge that might be investigated further. After describing the factors that influence run-time behavior, the analysis of test data reveals the potential effects of these factors and the extent to which they affect the final solution.

A large number of test measurements are recorded during the testing phase of the algorithms. However, three are distinguished by their demonstration of

unique characteristics supporting earlier claims regarding algorithm behavior.

The input files used during testing are slight revisions of a baseline input file[1]. This file defines a simple communication network consisting of eight nodes. The left-most nodes send messages to the right-most nodes using the center nodes for relay if necessary. An illustration is provided in Figure 6.2.

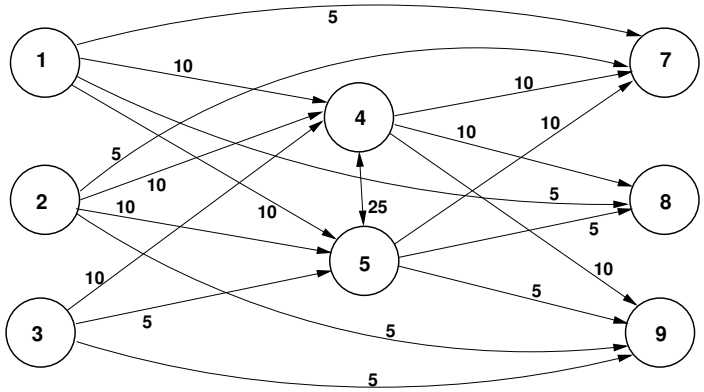

Figure 6.2: *The communication network used for testing*

The results presented in this section are all produced by making simple modifications to a single baseline input file. The significant affect that slight modifications have on algorithm run-time further supports the difficulty in predicting execution behavior for any single input. The intent of the analysis is to illuminate specific run-time characteristics and allow the user to make well-informed decisions about how to modify a mission plan to facilitate success.

With respect to the search space, a feasible solution space reflects one of four situations. Each has a unique effect on algorithm behavior. The situations are associated with identifiable consequences.

1. **The feasible solution space falls within PEA's search space.** When the feasible solution space is accessible by PEA, all three algorithms find a feasible solution. Each algorithm performs an identical set of scheduling decisions during execution due to the shared execution

[1]Appendix A contains all input files used to produce the results in this chapter

nucleus described in Chapter 5. Since additional backtracking is not required to find a feasible solution, both APEA and TEA determine the same feasible solution in the search time recorded for PEA.

2. **The feasible solution space falls within APEA's search space.** Under this condition, PEA fails to find a feasible solution while both APEA and TEA succeed. Since epoch backtracking is not required in order to find a feasible solution, TEA never utilizes this option during execution. Therefore, the execution times for both APEA and TEA are identical. The size and location of the feasible solution space, coupled with its distance from the initial point in APEA's search space, define the search time required to determine feasibility.

3. **The feasible solution space falls within TEA's search space.** In this case, TEA is the only algorithm able to find a feasible solution. Both PEA and APEA exhaust their search space and report infeasibility. Depending on input space parameters, execution times for APEA and TEA can be quite large.

4. **The feasible solution space is null.** In this case, all three algorithms fail to find a solution. Further, each evaluates all routing schedules contained in the respective search space. The exhaustive search can result in very high execution times for APEA and TEA depending on input space characteristics.

In the event that a feasible solution space falls on the border of a search space, an algorithm only considers points in the intersection of the two spaces. An algorithm employs additional backtracking options only when no further progress can be gained under the current constraints.

Table 6.1 presents the results of the three distinct tests selected for analysis. Each of the tests and their associated results are discussed in the sections that follow. The run-time results found in this section are not intended to be comprehensive. Rather, they are presented to illustrate some of the behavioral characteristics that characterize the intersection of the three algorithms.

Test	PEA		APEA		TEA	
	(seconds)	(result)	(seconds)	(result)	(seconds)	(result)
1	0.14	success	0.12	success	0.11	success
2	0.13	fail	0.14	fail	407.16	fail
3	0.21	fail	989.80	success	1028.14	success

Table 6.1: *Execution results selected for analysis*

6.2.1 Test 1

Test one defines a scenario in which all three algorithms find a feasible solution, indicating that the feasible solution spaces lies within the search space of PEA. For APEA and TEA, message or epoch backtracking is unnecessary to find a solution. Consequently, similar run-times are experienced since all three algorithms make the same decisions during execution.

PEA finds a feasible solution, implying that feasibility is fairly easy to establish. Execution results of this type usually indicate that the communication network is largely under-utilized. An intuitive conclusion is that excessive bandwidth is provided for the message requirements of the mission.

6.2.2 Test 2

No algorithm finds a feasible solution in this test, indicating that a feasible solution space for this mission plan does not exist. Consequently, each algorithm exhaustively evaluates its respective search space before terminating execution with an infeasible solution. Although no algorithm finds a feasible solution, valuable information is conveyed in these results.

In examining the differences in execution times among the algorithms, we see that very little difference exists between PEA and APEA. This situation may indicate an early establishment of infeasibility, leaving APEA few alternatives to examine before termination. Early detection of infeasibility is usually indicative of a substantial bottleneck in the communication network. Examination of the decisions made during execution may provide further insight.

TEA also fails to find a feasible solution, indicating that APEA is far from finding a feasible solution at the time of program termination. The joint outcomes suggest a significant lack of communication resources. Examination of the message assignment logs may provide insight into which messages are contributing to the infeasible solution.

6.2.3 Test 3

In the third test, both APEA and TEA find a feasible solution where PEA cannot. Several interesting observations are derived from the results.

The large difference in execution time between PEA and APEA show that an extensive amount of backtracking is required to find a feasible solution, indicating that available resources and communication requirements are very closely matched. If a bandwidth surplus is required for an emergency situation, the user may want to increase the communications bandwidth for the mission.

6.2.4 Behavioral Conclusions

During testing, we failed to create an input file in which both PEA and APEA failed while TEA succeeded. The inability to do so raised speculations about the relationships among the algorithms.

The two heuristics discussed in the previous section allow an algorithm to generate desirable routing schedules early in execution. It could be the case that after a certain point in execution, only undesirable routing schedules remain to be generated. At this instance, it may be more effective to stop execution prematurely and report that no reasonable feasible solution is available. This dividing line may separate APEA and TEA, where APEA generates the routing schedules that have the best chance of succeeding and leaves only undesirable routing schedules for TEA.

Determining that no feasible solution exists is costly, however, the first routing schedule generated by an algorithm is unlikely to result in a feasible solution unless the the resources far outweigh the communication requirements. The behavior of PEA is consistent with this claim. Given the results

recorded during testing, APEA seems to be the best balance of efficiency and effectiveness. In all tests performed, APEA never reports a false positive solution. The difference in execution time between APEA and TEA, however, is extraordinarily high. In fact, the need to employ total enumeration to find a feasible solution may suggest a demanding, if not unacceptable, level of available resources.

Interpreting run-time results can be difficult. Although examining the input space sheds little light on expected behavior, analyzing the results of a test run does provide valuable information.

6.3 The Effect of Path Precalculation

As described in the recursive framework, paths are constructed on the spot for each message during execution. This overhead can be eliminated by precalculating paths at the start of each epoch to relieve computation overhead during the epoch evaluation itself. This is accomplished by calculating every possible path for each source-destination pair prior to epoch evaluation and storing the paths in a linked list. This list is sorted by source node and then by destination node. It takes $O(2n)$ time to find a path in the list, which is less demanding than constructing the same path repeatedly during execution. The results of this heuristic on two of the tests presented in the previous section using TEA are shown in Table 6.2.

	Disabled (seconds)	**Enabled** (seconds)	**Time Reduction Factor**
Test 2	407.16	43.68	9.32
Test 3	1028.14	27.64	37.19

Table 6.2: *The run-time effects of the path precalculation heuristic*

These results show dramatic increases in efficiency for the same level of effectiveness. By using path precalculation, an algorithm significantly reduces the time required to cover its search space. The larger the search space, the greater run-time reduction achievable using this heuristic.

6.4 Summary

The dimensions of the search, solution, and feasible solution spaces are governed by the numerous multi-variable interactions among input parameters. Algorithm behavior is dependent on these dimensions; thus run-time behavior is very difficult to predict. Useful information can be gained through the analysis of execution results, allowing a user to modify a mission plan in order to meet necessary requirements.

The characteristics of a search space are influenced by a large number of factors. The average number of paths in a communication network, as well as the average number of messages that must be transmitted, play a key role in defining search space dimensions. Additionally, message window size and average message size also have an effect on search space dimensions.

Two heuristics, due-date ordering and work conserving scheduling, are used to improve the scheduling decisions of the three algorithms, resulting in increased efficiency during epoch evaluations and a greater chance of finding a feasible solution in a shorter amount of time.

Experimental results presented in Section 6.2 show various algorithm behaviors for different input scenarios. Testing results indicate that APEA offers the best balance of solution accuracy and run-time performance. Results presented show a few of the behavioral characteristics exhibited by the algorithms, although a more extensive inquiry is the subject of work to follow.

Finally, path precalculation is a heuristic that can potentially reduce algorithm execution time by extraordinary amounts. For the results presented in Section 6.2, we see a speed increase by a factor of 37 over the same algorithm not employing path-precalculation.

Chapter 7

Conclusions and Future Work

The goal of this research is to develop and implement an algorithm to solve the extended MCNF problem to provide decision support for mission planning. Three algorithms are presented, each with increasing capabilities, that can be used to achieve a solution. During development, several unanswered questions have emerged that remain topics for future work. The following sections provide a summary of the research reported in this thesis and outline areas of potential future endeavors.

7.1 Concluding Summary

The research reported in this thesis provides a solution to the extended MCNF problem by detailing a tiered execution model comprised of three algorithms. Unlike previous approaches, our solution takes into account the ability to define relay nodes within a communications network. These relay nodes function solely to transfer data from one node to another, enabling a mission planner to define realistic communication networks. Also unique to our solution is the concept of a time-sliced event model where each message is sent and received in a single time slice called an epoch. A mission is composed of a finite number of epochs and the primary goal is to meet the communication requirements without exceeding the available resources for each epoch.

Prior to our solution, the most common algorithm used to solve problems of this nature was NETFLO. The extension of the decision support context to include NTN:RADS produces requirements absent in the typical MCNF problem domain. While NETFLO could be sufficient in addressing bandwidth capacity, it is not easily adaptable to the real-time message delivery problem.

The MCNF problem is extended to focus on feasibility rather than optimality. Coupled with the notion of a time epoch and the need to support relay nodes, the definition of our problem is formed. By assuring communications feasibility, we are able to provide accurate decision support for mission planning. In addition, the algorithms developed can point out possible problem areas that might exist in a given network or set of message requirements.

Each of the three algorithms offer varying trade-offs between efficiency and effectiveness. PEA offers significant efficiency with the possibility of being unable to find a feasible solution depending on input. APEA offers the best balance between the two, while TEA provides definitive answers at the expense of potentially unacceptable solution times. These choices enable a user to weigh the importance of efficiency vs. effectiveness during algorithm selection.

Although providing varying degrees of functionality, each algorithm shares the same execution nucleus. This nucleus employs a recursive framework to take advantage of the run-time stack during execution. All internal data sets are stored as linked lists and messages are sorted according to earliest due date. Each epoch is evaluated recursively until a successful routing schedule is discovered or all possible solutions are exhausted.

The combination of algorithm with input data define unique search, solution, and feasible solution spaces. The dimensions of theses spaces can be used to predict algorithm behavior. However, due to the large number of multivariable interactions and the complex relationships among the variables, the search and solution space dimensions are nearly impossible to predict. Useful information can be gained through execution analysis that allows a user to modify existing input parameters to increase chances of success.

From among the numerous conceived heuristics, only three are selected for implementation and test. Results gathered during testing, show that both due-date ordering and work conserving scheduling offer significant decreases

in algorithm execution time while increasing the likelihood of finding a feasible solution. The third heuristic, path-precalculation, decreases execution time at a rate that is inversely proportional to input size.

In summary, the solution presented here solves the resource allocation problem for message delivery in a way that enables evaluation of real world communication scenarios. While our target application domain is tactical communication networks, the algorithms developed can be used for a wide variety of allocation problems outside of a military setting. For example, planning a network for a college campus or a wireless network in a metropolitan area.

7.2 Future Work

Although the primary goal of this research is attained, many questions remain to be addressed in extensions and future work. This sections outlines a few possibilities.

7.2.1 Usability

Usability requires that our algorithms be presented to the user with an intuitive graphical user interface (GUI). This interface should allow the user to manipulate the algorithms and view results in a way that is easy to understand and interpret. Future opportunities also include a dynamic network topology visualization that updates in real-time as messages are assigned within a network during execution. The possibilities are infinite, but a user friendly execution interface is required if wide-spread usage should ever occur.

7.2.2 Additional Heuristics

Most of the obvious heuristics found their way into our implementation. However, additional heuristics perhaps not as obvious were proposed throughout the course of this research. This section outlines the heuristics uncovered during research, but that remain unimplemented.

1. *Precalculate routing data at nodes.* Path computation time can be potentially decreased by allowing nodes to store connectivity information for use when making routing decisions. Existing Internet routing protocols implement a similar kind of routing information recording which could be useful to our algorithms as well. Pruning the path search space can reduce computation overhead significantly, especially for large network topologies.

2. *Relay node buffering.* This is commonly referred to as store–and–forward routing. By relaxing the assumption that a message must complete transmission in a single epoch, messages can be stored temporarily at relay nodes and continue transmission in successive epochs. This provides an algorithm with greater scheduling flexibility and may reduce the effort required to find a feasible solution. A potential drawback to this heuristic is that the overhead for message set maintenance can increase in complexity due to buffered messages having greater priority than non-buffered messages at the start of an epoch.

3. *Split messages across channels and epochs.* Some messages may be of awkward size and hinder an algorithm's ability to make optimal scheduling decisions. Messages of this type can be broken into multiple messages of smaller size in order to allow more effective utilization of communication resources. The potential for overall feasibility can also be improved by sending single messages along multiple paths, allowing a more evenly balanced load on network channels. The primary aim of this heuristic is to exploit epoch and channel assignment flexibility and to use remaining channel bandwidth more effectively. Should a message be split across an epoch boundary, the remaining portion must be sent first in the next epoch ahead of other messages.

4. *Aggressive front loading.* By default, an algorithm attempts to respect channel threshold if at all possible. Originally, channel threshold is only exceeded when necessary on a per-message basis. Aggressive front loading allows an algorithm to continue exceeding threshold once it has been exceeded the first time. The idea is that once threshold is exceeded, a feasible within channel threshold solution is impossible. It is therefore pointless to continue respecting channel threshold any longer. This can increase the odds of finding a feasible solution by continuing

to ignore threshold for the remaining evaluation. An algorithm can employ this heuristic on a per-path basis or on a global basis depending on the desired behavior of the algorithm. Although this heuristic can afford an algorithm greater flexibility in finding a feasible solution, it can also significantly increase the number of possible message combinations and thus dramatically increase required execution time.

5. *Epoch snapshots.* A snapshot of the messages available for sending can be recorded at the start of each epoch and later compared during reevaluations of that same epoch. If the snapshots match, the epoch need not be evaluated since no feasible routing schedule can be found. This has the potential to save computation overhead significantly in the event that multiple epoch reevaluations are taking place. However, the addition memory required to store these snapshots might make this heuristic inappropriate.

7.2.3 Extension from Feasibility to Optimality

The NTN:RADS problem is derived by extending the MCNF problem for real-time message delivery and focusing on feasibility rather than optimality. Given the solutions presented in this research, it seems a reasonable extension to re-introduce optimality into the requirement set.

7.3 Research Results: Availability and Access

Without access to real test data, the algorithm testing phase included input files thought to reasonable in expressing a set of mission requirements. Without the ability to compare these input files with real data, judging overall algorithm performance is difficult. This is an important extension left for future exploration.

Bibliography

[1] Gnu compiler collection. Website. http://gcc.gnu.org.

[2] The gnu compiler for the java programming language. Website. http://gcc.gnu.org/java/.

[3] Virginia Tech Systems Research Center. Website. http://www.src.vt.edu.

[4] R. K. Ahuja, T. L. Magnanti, and J. B. Orlin. *Network Flows: Theory, Algorithms, and Applications*. Prentice-Hall, Englewood Cliffs, NJ, 1993.

[5] J. P. Brumbaugh-Smith and D. R. Sheir. Minimax models for diverse routing. *INFORMS Journal on Computing*, 14(1):81–95, Winter 2002.

[6] J. Castro and N. Nabona. An implementation of linear and nonlinear multicommodity network flows. *European Journal of Operational Research*, 92:37–53, 1996.

[7] A. K. Cebrowski and J. J. Garstka. Network-centric warfare: Its origin and future. Website, 1998. http://www.usni.org/ Proceedings/Articles98/PROcebrowski.htm.

[8] R. R. Forgleman and S. E. Widnall. Cornerstones of information warfare. Website. http://www.af.mil/lib/corner.html.

[9] A. Frangioni and G. Gallo. A bundle type dual-ascent approach to linear multicommodity min-cost flow problems. *INFORMS Journal on Computing*, 11(4):370–393, Fall 1999.

[10] M. R. Garey and D. S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completness*. W. H. Freeman and Company, New York, 1979.

[11] K. Holmberg and J. Hellstrand. Solving the uncapacitated network design problem by a lagrangean heuristic and brand-and-bound. *Operations Research*, 46(2):247–259, 1998.

[12] K. Holmberg and D. Yuan. A lagrangian heuristic based branch-and-bound approach for the capacitated network design problem. *Operations Research*, 48(3):461–481, 2002.

[13] K. Holmberg and D. Yuan. A multicommodity network-flow problem with side constraints on paths solved by column generation. *INFORMS Journal on Computing*, 15(1):42–57, Winter 2003.

[14] J. L. Kennington. A survey of linear cost multicommodity network flows. *Operations Research*, 26(2):209–236, 1978.

[15] D. McPherson, R. E. Nance, and J. D. Arthur. Constructing an architectural database to support communications resource allocation. Research Report SRC-01-005, Virginia Polytechnic Institute and State University, Systems Research Center and Department of Computer Science, July 2001.

[16] R. E. Nance, J. D. Arthur, and D. McPherson. Communications resource allocation for the VON project C4I architecture. Research Report SRC-01-004, Virginia Polytechnic Institute and State University, Systems Research Center and Department of Computer Science, July 2001.

[17] K. D. Slaght. Network centric warfare from powerpoint to reality. Website. URL not available as of October 2003.

[18] J. R. Steele. Determining communications resource feasibility in a tactical communications network. Master's thesis, Virginia Polytechnic Institute and State University, April 2002.
http://scholar.lib.vt.edu/ theses/available/etd-05092002-152507/.

[19] A. R. Washburn. Bits, bangs or bucks? the coming information crisis. *Phalanx*, 34(3):200–201, September 2001.

Appendix A

Input Files Used for Testing

A.1 Test File 1

```
##############################################################################
#
# PROBLEM DEFINITION (pd)
#
# Note: Epochs begin counting from 0.
#
#      Name                  Number of Epochs   Epoch Size

  pd   s2_p1_c2                     4              1.0

##############################################################################
#
# PLATFORMS (pl)
#
# NOTE: Right now, a platform is simply an id number and a name.
#
# TODO: Names sould be delimited by quotes and not require underscores
#       for spaces.
#
#      ID   Name

  pl   1    platform_1
  pl   2    platform_2
  pl   3    platform_3
  pl   4    platform_4
  pl   5    platform_5
  pl   6    platform_6

##############################################################################
#
# ROLES (ro)
#
```

```
# NOTE: Right now, a role is simply an id number and a name.
#
#      ID   Name

  ro   1    role_send_1
  ro   2    role_send_2
  ro   3    role_send_3
  ro   4    role_recv_1
  ro   5    role_recv_2
  ro   6    role_recv_3

##############################################################################
#
# NODES (no)
#
#      ID   Platform   Role   Type         Begin   End

# send nodes
  no   1       1        1     Send           1      4
  no   2       2        2     Send           1      4
  no   3       3        3     Send           1      4

# receive nodes
  no   4       4        4     Receive        1      4
  no   5       5        5     Receive        1      4
  no   6       6        6     Receive        1      4

##############################################################################
#
# CHANNELS (ch)
#
# Note: Channels are bi-directional between nodes.
#
#      ID   From   To   Type   Bandwidth   Threshold   Cost   Begin   End

# channels out of node 1
  ch   1     1      4    Any       15          80         1       1     4
  ch   2     1      5    Any       7           80         1       1     4
  ch   3     1      6    Any       10          80         1       1     4

# channels out of node 2
  ch   4     2      4     Any       15         80         1       1     4
  ch   5     2      5     Any       20         80         1       1     4
  ch   6     2      6     Any       8          80         1       1     4

# channels out of node 3
  ch   7     3      5     Any       10         80         1       1     4
  ch   8     3      6     Any       20         80         1       1     4

##############################################################################
#
# MESSAGES (me)
#
#   From Node   To Node   Start Epoch   ID   Size (MB)   Due Epoch   Type

##############################################################################
# messages sent from node 1
```

```
##############################################################################

#   From Node   To Node   Start Epoch   ID   Size (MB)   Due Epoch   Type
  me    1          4          1        1       1            2      Any
  me    1          5          1        1       2            4      Any
  me    1          6          1        1       3            4      Any
  me    1          4          1        1       2            3      Any
  me    1          4          1        2       3            2      Any
  me    1          6          1        1       4            3      Any
  me    1          6          1        2       1            2      Any

#   From Node   To Node   Start Epoch   ID   Size (MB)   Due Epoch   Type

  me    1          5          2        1       1            4      Any
  me    1          5          2        2       2            3      Any
  me    1          5          2        3       3            4      Any
  me    1          5          2        4       5            4      Any
  me    1          5          2        5       1            4      Any

#   From Node   To Node   Start Epoch   ID   Size (MB)   Due Epoch   Type
  me    1          4          3        1       2            4      Any
  me    1          5          3        1       2            4      Any
  me    1          5          3        2       3            4      Any

##############################################################################
# messages sent from node 2
##############################################################################

#   From Node   To Node   Start Epoch   ID   Size (MB)   Due Epoch   Type
  me    2          4          1        1       3            2      Any
  me    2          5          1        1       2            4      Any
  me    2          6          1        1       1            4      Any
  me    2          4          1        1       2            2      Any

#   From Node   To Node   Start Epoch   ID   Size (MB)   Due Epoch   Type
  me    2          4          2        1       2            3      Any
  me    2          4          2        2       4            3      Any
  me    2          4          2        3       1            3      Any
  me    2          4          2        4       1            4      Any
  me    2          5          2        1       2            4      Any
  me    2          5          2        2       2            3      Any
  me    2          5          2        3       3            3      Any

#   From Node   To Node   Start Epoch   ID   Size (MB)   Due Epoch   Type
  me    2          6          3        1       3            4      Any
  me    2          6          3        2       1            4      Any
  me    2          6          3        3       3            4      Any
  me    2          4          3        1       2            4      Any

##############################################################################
# messages sent from node 3
##############################################################################

#   From Node   To Node   Start Epoch   ID   Size (MB)   Due Epoch   Type
  me    3          6          1        1       3            2      Any
  me    3          6          1        2       1            3      Any
  me    3          6          1        3       3            2      Any
```

```
  me    3          6          1        4       4            2      Any
  me    3          5          1        1       1            2      Any
  me    3          5          1        2       1            4      Any

#   From Node   To Node   Start Epoch   ID   Size (MB)   Due Epoch   Type
  me    3          6          2        1       4            3      Any
  me    3          6          2        2       3            3      Any
  me    3          5          2        1       4            3      Any
  me    3          5          2        2       5            4      Any
  me    3          5          2        3       1            4      Any
  me    3          5          2        4       6            3      Any

#   From Node   To Node   Start Epoch   ID   Size (MB)   Due Epoch   Type
  me    3          6          3        1       3            4      Any
  me    3          6          3        2       3            4      Any
  me    3          6          3        3       2            4      Any
```

A.2 Test File 2

```
##############################################################################
#
# PROBLEM DEFINITION (pd)
#
# Note: Epochs begin counting from 0.
#
# Name                   Number of Epochs   Epoch Size

  pd   s2_p1_c2                     4             1.0

##############################################################################
#
# PLATFORMS (pl)
#
# NOTE: Right now, a platform is simply an id number and a name.
#
# TODO: Names sould be delimited by quotes and not require underscores
#       for spaces.
#
#      ID   Name

  pl   1    platform_1
  pl   2    platform_2
  pl   3    platform_3
  pl   4    platform_4
  pl   5    platform_5
  pl   6    platform_6

##############################################################################
#
# ROLES (ro)
#
# NOTE: Right now, a role is simply an id number and a name.
#
#      ID   Name
```

```
  ro   1    role_send_1
  ro   2    role_send_2
  ro   3    role_send_3
  ro   4    role_recv_1
  ro   5    role_recv_2
  ro   6    role_recv_3

##############################################################################
#
# NODES (no)
#
#      ID   Platform   Role   Type         Begin   End

# send nodes
  no   1       1        1     Send           1      4
  no   2       2        2     Send           1      4
  no   3       3        3     Send           1      4

# receive nodes
  no   4       4        4     Receive        1      4
  no   5       5        5     Receive        1      4
  no   6       6        6     Receive        1      4

##############################################################################
#
# CHANNELS (ch)
#
# Note: Channels are bi-directional between nodes.
#
#      ID   From   To   Type   Bandwidth   Threshold   Cost   Begin   End

# channels out of node 1
  ch   1     1      4    Any       15          80        1      1      4
  ch   2     1      5    Any       7           80        1      1      4
  ch   3     1      6    Any       10          80        1      1      4

# channels out of node 2
  ch   4     2      4    Any       15          80        1      1      4
  ch   5     2      5    Any       20          80        1      1      4
  ch   6     2      6    Any       8           80        1      1      4

# channels out of node 3
  ch   7     3      5    Any       10          80        1      1      4
  ch   8     3      6    Any       20          80        1      1      4

##############################################################################
#
# MESSAGES (me)
#
#   From Node   To Node   Start Epoch   ID   Size (MB)   Due Epoch   Type

##############################################################################
# messages sent from node 1
##############################################################################

#   From Node   To Node   Start Epoch   ID   Size (MB)   Due Epoch   Type
```

```
  me    1          4          1        1       1            2      Any
  me    1          5          1        1       2            4      Any
  me    1          6          1        1       3            4      Any
  me    1          4          1        1       2            3      Any
  me    1          4          1        2       3            2      Any
  me    1          6          1        1       4            3      Any
  me    1          6          1        2       1            2      Any

#   From Node   To Node   Start Epoch   ID   Size (MB)   Due Epoch   Type

  me    1          5          2        1       1            4      Any
  me    1          5          2        2       2            3      Any
  me    1          5          2        3       3            4      Any
  me    1          5          1        4       5            4      Any
  me    1          5          2        5       1            4      Any

#   From Node   To Node   Start Epoch   ID   Size (MB)   Due Epoch   Type
  me    1          4          3        1       2            4      Any
  me    1          5          3        1       2            4      Any
  me    1          5          3        2       3            4      Any

##############################################################################
# messages sent from node 2
##############################################################################

#   From Node   To Node   Start Epoch   ID   Size (MB)   Due Epoch   Type
  me    2          4          1        1       3            2      Any
  me    2          5          1        1       2            4      Any
  me    2          6          1        1       1            4      Any
  me    2          4          1        1       2            2      Any

#   From Node   To Node   Start Epoch   ID   Size (MB)   Due Epoch   Type
  me    2          4          2        1       2            3      Any
  me    2          4          1        2       4            3      Any
  me    2          4          2        3       1            3      Any
  me    2          4          2        4       1            4      Any
  me    2          5          2        1       2            4      Any
  me    2          5          2        2       2            3      Any
  me    2          5          2        3       3            3      Any

#   From Node   To Node   Start Epoch   ID   Size (MB)   Due Epoch   Type
  me    2          6          3        1       3            4      Any
  me    2          6          3        2       1            4      Any
  me    2          6          3        3       3            4      Any
  me    2          4          3        1       2            4      Any

##############################################################################
# messages sent from node 3
##############################################################################

#   From Node   To Node   Start Epoch   ID   Size (MB)   Due Epoch   Type
  me    3          6          1        1       3            2      Any
  me    3          6          1        2       1            3      Any
  me    3          6          1        3       3            2      Any
  me    3          6          1        4       4            2      Any
  me    3          5          1        1       1            2      Any
  me    3          5          1        2       1            4      Any
```

```
#   From Node   To Node   Start Epoch   ID   Size (MB)   Due Epoch   Type
  me    3          6           2        1       4            3      Any
  me    3          6           2        2       3            3      Any
  me    3          5           2        1       4            3      Any
  me    3          5           2        2       5            4      Any
  me    3          5           2        3       1            4      Any
  me    3          5           2        4       6            3      Any

#   From Node   To Node   Start Epoch   ID   Size (MB)   Due Epoch   Type
  me    3          6           3        1       3            4      Any
  me    3          6           3        2       3            4      Any
  me    3          6           3        3       2            4      Any
```

A.3 Test File 3

```
###############################################################################
#
# PROBLEM DEFINITION (pd)
#
# Note: Epochs begin counting from 0.
#
#      Name                   Number of Epochs   Epoch Size

  pd   s2_p1_c2                     4              1.0

###############################################################################
#
# PLATFORMS (pl)
#
# NOTE: Right now, a platform is simply an id number and a name.
#
# TODO: Names sould be delimited by quotes and not require underscores
#       for spaces.
#
#      ID   Name

  pl   1    platform_1
  pl   2    platform_2
  pl   3    platform_3
  pl   4    platform_4
  pl   5    platform_5
  pl   6    platform_6

###############################################################################
#
# ROLES (ro)
#
# NOTE: Right now, a role is simply an id number and a name.
#
#      ID   Name

  ro   1    role_send_1
  ro   2    role_send_2
```

```
  ro   3    role_send_3
  ro   4    role_recv_1
  ro   5    role_recv_2
  ro   6    role_recv_3

##############################################################################
#
# NODES (no)
#
#      ID   Platform   Role   Type         Begin   End

# send nodes
  no   1       1        1     Send           1      4
  no   2       2        2     Send           1      4
  no   3       3        3     Send           1      4

# receive nodes
  no   4       4        4     Receive        1      4
  no   5       5        5     Receive        1      4
  no   6       6        6     Receive        1      4

##############################################################################
#
# CHANNELS (ch)
#
# Note: Channels are bi-directional between nodes.
#
#      ID   From   To   Type   Bandwidth   Threshold   Cost   Begin   End

# channels out of node 1
  ch   1     1      4    Any       15          80       1       1      4
  ch   2     1      5    Any       7           80       1       1      4
  ch   3     1      6    Any       10          80       1       1      4

# channels out of node 2
  ch   4       2    4    Any       15          80       1       1      4
  ch   5       2    5    Any       20          80       1       1      4
  ch   6       2    6    Any       8           80       1       1      4

# channels out of node 3
  ch   7       3    5    Any       10          80       1       1      4
  ch   8       3    6    Any       20          80       1       1      4

##############################################################################
#
# MESSAGES (me)
#
#   From Node   To Node   Start Epoch   ID   Size (MB)   Due Epoch   Type

##############################################################################
# messages sent from node 1
##############################################################################

#   From Node   To Node   Start Epoch   ID   Size (MB)   Due Epoch   Type
  me    1          4          1        1       1            2      Any
  me    1          5          1        1       2            4      Any
  me    1          6          1        1       3            4      Any
```

```
  me    1          4          1        1       2            3      Any
  me    1          4          1        2       3            2      Any
  me    1          6          1        1       4            3      Any
  me    1          6          1        2       1            2      Any

#   From Node   To Node   Start Epoch   ID   Size (MB)   Due Epoch   Type

  me    1          5          2        1       1            4      Any
  me    1          5          2        2       2            3      Any
  me    1          5          2        3       3            4      Any
  me    1          5          1        4       5            4      Any
  me    1          5          2        5       1            4      Any

#   From Node   To Node   Start Epoch   ID   Size (MB)   Due Epoch   Type
  me    1          4          3        1       2            4      Any
  me    1          5          3        1       2            4      Any
  me    1          5          3        2       3            4      Any

###############################################################################
# messages sent from node 2
###############################################################################

#   From Node   To Node   Start Epoch   ID   Size (MB)   Due Epoch   Type
  me    2          4          1        1       3            2      Any
  me    2          5          1        1       2            4      Any
  me    2          6          1        1       1            4      Any
  me    2          4          1        1       2            2      Any

#   From Node   To Node   Start Epoch   ID   Size (MB)   Due Epoch   Type
  me    2          4          2        1       2            3      Any
  me    2          4          2        2       4            3      Any
  me    2          4          2        3       1            3      Any
  me    2          4          2        4       1            4      Any
  me    2          5          2        1       2            4      Any
  me    2          5          2        2       2            3      Any
  me    2          5          2        3       3            3      Any

#   From Node   To Node   Start Epoch   ID   Size (MB)   Due Epoch   Type
  me    2          6          3        1       3            4      Any
  me    2          6          3        2       1            4      Any
  me    2          6          3        3       3            4      Any
  me    2          4          3        1       2            4      Any

###############################################################################
# messages sent from node 3
###############################################################################

#   From Node   To Node   Start Epoch   ID   Size (MB)   Due Epoch   Type
  me    3          6          1        1       3            2      Any
  me    3          6          1        2       1            3      Any
  me    3          6          1        3       3            2      Any
  me    3          6          1        4       4            2      Any
  me    3          5          1        1       1            2      Any
  me    3          5          1        2       1            4      Any

#   From Node   To Node   Start Epoch   ID   Size (MB)   Due Epoch   Type
  me    3          6          2        1       4            3      Any
```

```
  me    3          6          2        2       3             3      Any
  me    3          5          2        1       4             3      Any
  me    3          5          2        2       5             4      Any
  me    3          5          2        3       1             4      Any
  me    3          5          2        4       6             3      Any

#   From Node   To Node   Start Epoch   ID   Size (MB)   Due Epoch   Type
  me    3          6          3        1       3             4      Any
  me    3          6          3        2       3             4      Any
  me    3          6          3        3       2             4      Any
```

A.4 Test File 4

```
###############################################################################
#
# PROBLEM DEFINITION (pd)
#
# Note: Epochs begin counting from 0.
#
#      Name                   Number of Epochs   Epoch Size

  pd   s2_p1_c2                     4              1.0

###############################################################################
#
# PLATFORMS (pl)
#
# NOTE: Right now, a platform is simply an id number and a name.
#
# TODO: Names sould be delimited by quotes and not require underscores
#       for spaces.
#
#      ID   Name

  pl   1    platform_1
  pl   2    platform_2
  pl   3    platform_3
  pl   4    platform_4
  pl   5    platform_5
  pl   6    platform_6

###############################################################################
#
# ROLES (ro)
#
# NOTE: Right now, a role is simply an id number and a name.
#
#      ID   Name

  ro   1    role_send_1
  ro   2    role_send_2
  ro   3    role_send_3
  ro   4    role_recv_1
  ro   5    role_recv_2
```

```
  ro   6    role_recv_3

###############################################################################
#
# NODES (no)
#
#      ID   Platform   Role   Type         Begin   End

# send nodes
  no   1       1        1     Send           1      4
  no   2       2        2     Send           1      4
  no   3       3        3     Send           1      4

# receive nodes
  no   4       4        4     Receive        1      4
  no   5       5        5     Receive        1      4
  no   6       6        6     Receive        1      4

###############################################################################
#
# CHANNELS (ch)
#
# Note: Channels are bi-directional between nodes.
#
#      ID   From   To   Type   Bandwidth   Threshold   Cost   Begin   End

# channels out of node 1
  ch   1     1      4    Any       15          80       1       1      4
  ch   2     1      5    Any       7           80       1       1      4
  ch   3     1      6    Any       10          80       1       1      4

# channels out of node 2
  ch   4       2    4    Any       15          80       1       1      4
  ch   5       2    5    Any       20          80       1       1      4
  ch   6       2    6    Any       8           80       1       1      4

# channels out of node 3
  ch   7       3    5    Any       10          80       1       1      4
  ch   8       3    6    Any       20          80       1       1      4

###############################################################################
#
# MESSAGES (me)
#
#   From Node   To Node   Start Epoch   ID   Size (MB)   Due Epoch   Type

###############################################################################
# messages sent from node 1
###############################################################################

#   From Node   To Node   Start Epoch   ID   Size (MB)   Due Epoch   Type
  me    1          4          1        1       1           2      Any
  me    1          5          1        1       2           4      Any
  me    1          6          1        1       3           4      Any
  me    1          4          1        1       2           3      Any
  me    1          4          1        2       3           2      Any
  me    1          6          1        1       4           3      Any
```

```
  me    1          6          1        2       1             2      Any

#   From Node   To Node   Start Epoch   ID   Size (MB)   Due Epoch   Type

  me    1          5          2        1       1             4      Any
  me    1          5          2        2       2             3      Any
  me    1          5          2        3       3             4      Any
  me    1          5          2        4       5             4      Any
  me    1          5          2        5       1             4      Any

#   From Node   To Node   Start Epoch   ID   Size (MB)   Due Epoch   Type
  me    1          4          3        1       2             4      Any
  me    1          5          3        1       2             4      Any
  me    1          5          3        2       3             4      Any

###############################################################################
# messages sent from node 2
###############################################################################

#   From Node   To Node   Start Epoch   ID   Size (MB)   Due Epoch   Type
  me    2          4          1        1       3             2      Any
  me    2          5          1        1       2             4      Any
  me    2          6          2        1       1             4      Any
  me    2          4          1        1       2             2      Any

#   From Node   To Node   Start Epoch   ID   Size (MB)   Due Epoch   Type
  me    2          4          2        1       2             3      Any
  me    2          4          2        2       4             3      Any
  me    2          4          2        3       1             3      Any
  me    2          4          2        4       1             4      Any
  me    2          5          2        1       2             4      Any
  me    2          5          2        2       2             3      Any
  me    2          5          2        3       3             3      Any

#   From Node   To Node   Start Epoch   ID   Size (MB)   Due Epoch   Type
  me    2          6          3        1       3             4      Any
  me    2          6          3        2       1             4      Any
  me    2          6          3        3       3             4      Any
  me    2          4          3        1       2             4      Any

###############################################################################
# messages sent from node 3
###############################################################################

#   From Node   To Node   Start Epoch   ID   Size (MB)   Due Epoch   Type
  me    3          6          1        1       3             2      Any
  me    3          6          1        2       1             3      Any
  me    3          6          1        3       3             2      Any
  me    3          6          1        4       4             2      Any
  me    3          5          1        1       1             2      Any
  me    3          5          1        2       1             4      Any

#   From Node   To Node   Start Epoch   ID   Size (MB)   Due Epoch   Type
  me    3          6          2        1       4             3      Any
  me    3          6          2        2       3             3      Any
  me    3          5          2        1       4             3      Any
  me    3          5          2        2       5             4      Any
```

```
  me    3          5           2        3       1             4      Any
  me    3          5           2        4       6             3      Any

#   From Node   To Node   Start Epoch   ID   Size (MB)   Due Epoch   Type
  me    3          6           3        1       3             4      Any
  me    3          6           3        2       3             4      Any
  me    3          6           3        3       2             4      Any
```

A.5 Test File 5

```
################################################################################
#
# PROBLEM DEFINITION (pd)
#
# Note: Epochs begin counting from 0.
#
#      Name                   Number of Epochs   Epoch Size

  pd   s2_p1_c2                     4              1.0

################################################################################
#
# PLATFORMS (pl)
#
# NOTE: Right now, a platform is simply an id number and a name.
#
# TODO: Names sould be delimited by quotes and not require underscores
#       for spaces.
#
#      ID   Name

  pl   1    platform_1
  pl   2    platform_2
  pl   3    platform_3
  pl   4    platform_4
  pl   5    platform_5
  pl   6    platform_6

################################################################################
#
# ROLES (ro)
#
# NOTE: Right now, a role is simply an id number and a name.
#
#      ID   Name

  ro   1    role_send_1
  ro   2    role_send_2
  ro   3    role_send_3
  ro   4    role_recv_1
  ro   5    role_recv_2
  ro   6    role_recv_3

################################################################################
```

```
#
# NODES (no)
#
#      ID   Platform   Role   Type         Begin   End

# send nodes
  no   1       1        1     Send           1      4
  no   2       2        2     Send           1      4
  no   3       3        3     Send           1      4

# receive nodes
  no   4       4        4     Receive        1      4
  no   5       5        5     Receive        1      4
  no   6       6        6     Receive        1      4

###############################################################################
#
# CHANNELS (ch)
#
# Note: Channels are bi-directional between nodes.
#
#      ID   From   To   Type   Bandwidth   Threshold   Cost   Begin   End

# channels out of node 1
  ch   1     1      4    Any       15          80       1       1      4
  ch   2     1      5    Any       7           80       1       1      4
  ch   3     1      6    Any       10          80       1       1      4

# channels out of node 2
  ch   4       2    4    Any       15          80       1       1      4
  ch   5       2    5    Any       20          80       1       1      4
  ch   6       2    6    Any       8           80       1       1      4

# channels out of node 3
  ch   7       3    5    Any       10          80       1       1      4
  ch   8       3    6    Any       20          80       1       1      4

###############################################################################
#
# MESSAGES (me)
#
#   From Node   To Node   Start Epoch   ID   Size (MB)   Due Epoch   Type

###############################################################################
# messages sent from node 1
###############################################################################

#   From Node   To Node   Start Epoch   ID   Size (MB)   Due Epoch   Type
  me    1          4          1        1       1            2      Any
  me    1          5          1        1       2            4      Any
  me    1          6          1        1       3            4      Any
  me    1          4          1        1       2            3      Any
  me    1          4          1        2       3            2      Any
  me    1          6          1        1       4            3      Any
  me    1          6          1        2       1            2      Any

#   From Node   To Node   Start Epoch   ID   Size (MB)   Due Epoch   Type
```

```
  me    1          5          2        1       1            4      Any
  me    1          5          2        2       2            3      Any
  me    1          5          2        3       3            4      Any
  me    1          5          2        4       5            4      Any

#   From Node   To Node   Start Epoch   ID   Size (MB)   Due Epoch   Type
  me    1          4          3        1       2            4      Any
  me    1          5          3        1       2            4      Any
  me    1          5          3        2       3            4      Any

##############################################################################
# messages sent from node 2
##############################################################################

#   From Node   To Node   Start Epoch   ID   Size (MB)   Due Epoch   Type
  me    2          4          1        1       3            2      Any
  me    2          5          1        1       2            4      Any
  me    2          6          1        1       1            4      Any
  me    2          4          1        1       2            2      Any

#   From Node   To Node   Start Epoch   ID   Size (MB)   Due Epoch   Type
  me    2          4          2        1       2            3      Any
  me    2          4          2        2       4            3      Any
  me    2          4          2        3       1            3      Any
  me    2          4          2        4       1            4      Any
  me    2          5          2        1       2            4      Any
  me    2          5          2        2       2            3      Any
  me    2          5          2        3       3            3      Any

#   From Node   To Node   Start Epoch   ID   Size (MB)   Due Epoch   Type
  me    2          6          3        1       3            4      Any
  me    2          6          3        2       1            4      Any
  me    2          6          3        3       3            4      Any
  me    2          4          3        1       2            4      Any

##############################################################################
# messages sent from node 3
##############################################################################

#   From Node   To Node   Start Epoch   ID   Size (MB)   Due Epoch   Type
  me    3          6          1        1       3            2      Any
  me    3          6          1        2       1            3      Any
  me    3          6          1        3       3            2      Any
  me    3          6          1        4       4            2      Any
  me    3          5          1        1       1            2      Any
  me    3          5          1        2       1            4      Any

#   From Node   To Node   Start Epoch   ID   Size (MB)   Due Epoch   Type
  me    3          6          2        1       4            3      Any
  me    3          6          2        2       3            3      Any
  me    3          5          2        1       4            3      Any
  me    3          5          2        2       5            4      Any
  me    3          5          2        3       1            4      Any
  me    3          5          2        4       6            3      Any

#   From Node   To Node   Start Epoch   ID   Size (MB)   Due Epoch   Type
```

```
  me    3         6          3       1       3           4      Any
  me    3         6          3       2       3           4      Any
  me    3         6          3       3       2           4      Any
```

A.6 Test File 6

```
################################################################################
#
# PROBLEM DEFINITION (pd)
#
# Note: Epochs begin counting from 0.
#
#      Name                  Number of Epochs   Epoch Size

  pd   s2_p1_c2                     4              1.0

################################################################################
#
# PLATFORMS (pl)
#
# NOTE: Right now, a platform is simply an id number and a name.
#
# TODO: Names sould be delimited by quotes and not require underscores
#       for spaces.
#
#      ID   Name

  pl   1    platform_1
  pl   2    platform_2
  pl   3    platform_3
  pl   4    platform_4
  pl   5    platform_5
  pl   6    platform_6

################################################################################
#
# ROLES (ro)
#
# NOTE: Right now, a role is simply an id number and a name.
#
#      ID   Name

  ro   1    role_send_1
  ro   2    role_send_2
  ro   3    role_send_3
  ro   4    role_recv_1
  ro   5    role_recv_2
  ro   6    role_recv_3

################################################################################
#
# NODES (no)
#
#      ID   Platform   Role   Type         Begin   End
```

```
# send nodes
  no   1       1        1     Send          1      4
  no   2       2        2     Send          1      4
  no   3       3        3     Send          1      4

# receive nodes
  no   4       4        4     Receive       1      4
  no   5       5        5     Receive       1      4
  no   6       6        6     Receive       1      4

##############################################################################
#
# CHANNELS (ch)
#
# Note: Channels are bi-directional between nodes.
#
#      ID   From   To   Type   Bandwidth   Threshold   Cost   Begin   End

# channels out of node 1
  ch   1     1      4    Any       15          80       1       1      4
  ch   2     1      5    Any       7           80       1       1      4
  ch   3     1      6    Any       10          80       1       1      4

# channels out of node 2
  ch   4       2    4    Any       15          80       1       1      4
  ch   5       2    5    Any       20          80       1       1      4
  ch   6       2    6    Any       8           80       1       1      4

# channels out of node 3
  ch   7       3    5    Any       10          80       1       1      4
  ch   8       3    6    Any       20          80       1       1      4

##############################################################################
#
# MESSAGES (me)
#
#   From Node   To Node   Start Epoch   ID   Size (MB)   Due Epoch   Type

##############################################################################
# messages sent from node 1
##############################################################################

#   From Node   To Node   Start Epoch   ID   Size (MB)   Due Epoch   Type
  me    1          4          1        1       1            2      Any
  me    1          5          1        1       2            4      Any
  me    1          6          2        1       3            4      Any
  me    1          4          1        1       2            3      Any
  me    1          4          1        2       3            2      Any
  me    1          6          1        1       4            3      Any
  me    1          6          1        2       1            2      Any

#   From Node   To Node   Start Epoch   ID   Size (MB)   Due Epoch   Type

  me    1          5          2        1       1            4      Any
  me    1          5          2        2       2            3      Any
  me    1          5          2        3       3            4      Any
```

```
  me    1          5          2        4       5            4      Any
  me    1          5          2        5       1            4      Any

#   From Node   To Node   Start Epoch   ID   Size (MB)   Due Epoch   Type
  me    1          4          3        1       2            4      Any
  me    1          5          3        1       2            4      Any
  me    1          5          3        2       3            4      Any

###############################################################################
# messages sent from node 2
###############################################################################

#   From Node   To Node   Start Epoch   ID   Size (MB)   Due Epoch   Type
  me    2          4          1        1       3            2      Any
  me    2          5          1        1       2            4      Any
  me    2          6          1        1       1            4      Any
  me    2          4          1        1       2            2      Any

#   From Node   To Node   Start Epoch   ID   Size (MB)   Due Epoch   Type
  me    2          4          2        1       2            3      Any
  me    2          4          2        2       4            3      Any
  me    2          4          2        3       1            3      Any
  me    2          4          2        4       1            4      Any
  me    2          5          2        1       2            4      Any
  me    2          5          2        2       2            3      Any
  me    2          5          2        3       3            3      Any

#   From Node   To Node   Start Epoch   ID   Size (MB)   Due Epoch   Type
  me    2          6          3        1       3            4      Any
  me    2          6          3        2       1            4      Any
  me    2          6          3        3       3            4      Any
  me    2          4          3        1       2            4      Any

###############################################################################
# messages sent from node 3
###############################################################################

#   From Node   To Node   Start Epoch   ID   Size (MB)   Due Epoch   Type
  me    3          6          1        1       3            2      Any
  me    3          6          1        2       1            3      Any
  me    3          6          1        3       3            2      Any
  me    3          6          1        4       4            2      Any
  me    3          5          1        1       1            2      Any
  me    3          5          1        2       1            4      Any

#   From Node   To Node   Start Epoch   ID   Size (MB)   Due Epoch   Type
  me    3          6          2        1       4            3      Any
  me    3          6          2        2       3            3      Any
  me    3          5          2        1       4            3      Any
  me    3          5          2        2       5            4      Any
  me    3          5          2        3       1            4      Any
  me    3          5          2        4       6            3      Any

#   From Node   To Node   Start Epoch   ID   Size (MB)   Due Epoch   Type
  me    3          6          3        1       3            4      Any
  me    3          6          3        2       3            4      Any
  me    3          6          3        3       2            4      Any
```

A.7 Test File 7

```
################################################################################
#
# PROBLEM DEFINITION (pd)
#
# Note: Epochs begin counting from 0.
#
#      Name                    Number of Epochs   Epoch Size

  pd   s2_p1_c2                      4              1.0

################################################################################
#
# PLATFORMS (pl)
#
# NOTE: Right now, a platform is simply an id number and a name.
#
# TODO: Names sould be delimited by quotes and not require underscores
#       for spaces.
#
#      ID   Name

  pl   1    platform_1
  pl   2    platform_2
  pl   3    platform_3
  pl   4    platform_4
  pl   5    platform_5
  pl   6    platform_6

################################################################################
#
# ROLES (ro)
#
# NOTE: Right now, a role is simply an id number and a name.
#
#      ID   Name

  ro   1    role_send_1
  ro   2    role_send_2
  ro   3    role_send_3
  ro   4    role_recv_1
  ro   5    role_recv_2
  ro   6    role_recv_3

################################################################################
#
# NODES (no)
#
#      ID   Platform   Role   Type         Begin   End

# send nodes
  no   1       1        1     Send           1      4
  no   2       2        2     Send           1      4
  no   3       3        3     Send           1      4
```

```
# receive nodes
  no   4       4        4     Receive        1      4
  no   5       5        5     Receive        1      4
  no   6       6        6     Receive        1      4

###############################################################################
#
# CHANNELS (ch)
#
# Note: Channels are bi-directional between nodes.
#
#      ID   From   To   Type   Bandwidth   Threshold   Cost   Begin   End

# channels out of node 1
  ch   1     1      4    Any       15          80         1       1     4
  ch   2     1      5    Any       7           80         1       1     4
  ch   3     1      6    Any       10          80         1       1     4

# channels out of node 2
  ch   4     2      4     Any       15         80         1       1     4
  ch   5     2      5     Any       20         80         1       1     4
  ch   6     2      6     Any       8          80         1       1     4

# channels out of node 3
  ch   7     3      5     Any       10         80         1       1     4
  ch   8     3      6     Any       20         80         1       1     4

###############################################################################
#
# MESSAGES (me)
#
#   From Node   To Node   Start Epoch   ID   Size (MB)   Due Epoch   Type

###############################################################################
# messages sent from node 1
###############################################################################

#   From Node   To Node   Start Epoch   ID   Size (MB)   Due Epoch   Type
  me    1          4           1        1       1            2      Any
  me    1          5           1        1       2            4      Any
  me    1          6           3        1       3            4      Any
  me    1          4           1        1       2            3      Any
  me    1          4           1        2       3            2      Any
  me    1          6           1        1       4            3      Any
  me    1          6           1        2       1            2      Any

#   From Node   To Node   Start Epoch   ID   Size (MB)   Due Epoch   Type

  me    1          5           2        1       1            4      Any
  me    1          5           2        2       2            3      Any
  me    1          5           2        3       3            4      Any
  me    1          5           2        4       5            4      Any
  me    1          5           2        5       1            4      Any

#   From Node   To Node   Start Epoch   ID   Size (MB)   Due Epoch   Type
  me    1          4           3        1       2            4      Any
  me    1          5           3        1       2            4      Any
```

```
  me    1          5          3        2       3             4      Any

###############################################################################
# messages sent from node 2
###############################################################################

#   From Node   To Node   Start Epoch   ID   Size (MB)   Due Epoch   Type
  me    2          4          1        1       3             2      Any
  me    2          5          1        1       2             4      Any
  me    2          6          1        1       1             4      Any
  me    2          4          1        1       2             2      Any

#   From Node   To Node   Start Epoch   ID   Size (MB)   Due Epoch   Type
  me    2          4          2        1       2             3      Any
  me    2          4          2        2       4             3      Any
  me    2          4          2        3       1             3      Any
  me    2          4          2        4       1             4      Any
  me    2          5          2        1       2             4      Any
  me    2          5          2        2       2             3      Any
  me    2          5          2        3       3             3      Any

#   From Node   To Node   Start Epoch   ID   Size (MB)   Due Epoch   Type
  me    2          6          3        1       3             4      Any
  me    2          6          3        2       1             4      Any
  me    2          6          3        3       3             4      Any
  me    2          4          3        1       2             4      Any

###############################################################################
# messages sent from node 3
###############################################################################

#   From Node   To Node   Start Epoch   ID   Size (MB)   Due Epoch   Type
  me    3          6          1        1       3             2      Any
  me    3          6          1        2       1             3      Any
  me    3          6          1        3       3             2      Any
  me    3          6          1        4       4             2      Any
  me    3          5          1        1       1             2      Any
  me    3          5          1        2       1             4      Any

#   From Node   To Node   Start Epoch   ID   Size (MB)   Due Epoch   Type
  me    3          6          2        1       4             3      Any
  me    3          6          2        2       3             3      Any
  me    3          5          2        1       4             3      Any
  me    3          5          2        2       5             4      Any
  me    3          5          2        3       1             4      Any
  me    3          5          2        4       6             3      Any

#   From Node   To Node   Start Epoch   ID   Size (MB)   Due Epoch   Type
  me    3          6          3        1       3             4      Any
  me    3          6          3        2       3             4      Any
  me    3          6          3        3       2             4      Any
```

A.8 Test File 8

```
###############################################################################
```

```
#
# PROBLEM DEFINITION (pd)
#
# Note: Epochs begin counting from 0.
#
#      Name                   Number of Epochs   Epoch Size

  pd   s1_p1_c1                      10              1.0

################################################################################
#
# PLATFORMS (pl)
#
# NOTE: Right now, a platform is simply an id number and a name.
#
# TODO: Names sould be delimited by quotes and not require underscores
#       for spaces.
#
#      ID   Name

  pl   1    platform_1
  pl   2    platform_2
  pl   3    platform_3
  pl   4    platform_4
  pl   5    platform_5
  pl   6    platform_6
  pl   7    satelite_1
  pl   8    satelite_2

################################################################################
#
# ROLES (ro)
#
# NOTE: Right now, a role is simply an id number and a name.
#
#      ID   Name

  ro   1    role_send_1
  ro   2    role_send_2
  ro   3    role_send_3
  ro   4    role_recv_1
  ro   5    role_recv_2
  ro   6    role_recv_3
  ro   7    role_relay_1
  ro   8    role_relay_2

################################################################################
#
# NODES (no)
#
#      ID   Platform   Role   Type         Begin   End

# send nodes
  no   1       1        1     Send           1      10
  no   2       2        2     Send           1      10
  no   3       3        3     Send           1      10
```

```
# relay nodes
  no   4       7        7     Relay          1      10
  no   5       8        8     Relay          1      10

# receive nodes
  no   6       4        4     Receive        1      10
  no   7       5        5     Receive        1      10
  no   8       6        6     Receive        1      10

###############################################################################
#
# CHANNELS (ch)
#
# Note: Channels are bi-directional between nodes.
#
#      ID   From   To   Type   Bandwidth   Threshold   Cost   Begin   End

# channels out of node 1
  ch   1     1      8    Any       5           80        1      1     10
  ch   2     1      4    Any       10          80        1      1     10
  ch   3     1      7    Any       5           80        1      1     10
  ch   4     1      5    Any       10          80        1      1     10

# channels out of node 2
  ch   5     2      8    Any       5           80        1      1     10
  ch   6     2      4    Any       10          80        1      1     10
  ch   7     2      5    Any       10          80        1      1     10
  ch   8     2      6    Any       5           80        1      1     10

# channels out of node 3
  ch   9     3      4    Any       10          80        1      1     10
  ch   10    3      5    Any       5           80        1      1     10
  ch   11    3      6    Any       5           80        1      1     10

# channels out of node 4
  ch   12    4      5    Any       25          80        1      1     10
  ch   13    4      6    Any       10          80        1      1     10
  ch   14    4      7    Any       10          80        1      1     10
  ch   15    4      8    Any       10          80        1      1     10

# channels out of node 5
  ch   16    5      6    Any       5           80        1      1     10
  ch   17    5      7    Any       5           80        1      1     10
  ch   18    5      8    Any       10          80        1      1     10

###############################################################################
#
# MESSAGES (me)
#
#   From Node   To Node   Start Epoch   ID   Size (MB)   Due Epoch   Type

###############################################################################
# messages sent from node 1
###############################################################################

#   From Node   To Node   Start Epoch   ID   Size (MB)   Due Epoch   Type
  me    1          6           1        1       2            2      Any
```

```
  me    1          7          1        1       3            3      Any
  me    1          8          1        1       2            2      Any

#   From Node   To Node   Start Epoch   ID   Size (MB)   Due Epoch   Type
  me    1          6          2        1       2            3      Any
  me    1          6          2        2       3            4      Any
  me    1          7          2        1       2            3      Any
  me    1          7          2        2       3            4      Any
  me    1          7          2        3       1            5      Any
  me    1          8          2        1       2            3      Any
  me    1          8          2        2       3            4      Any

#   From Node   To Node   Start Epoch   ID   Size (MB)   Due Epoch   Type
  me    1          6          3        1       4            6      Any
  me    1          6          3        2       5            7      Any
  me    1          6          3        3       4            5      Any
  me    1          7          3        1       3            4      Any
  me    1          7          3        2       2            5      Any
  me    1          8          3        1       3            4      Any
  me    1          8          3        2       3            4      Any
  me    1          8          3        3       4            5      Any

#   From Node   To Node   Start Epoch   ID   Size (MB)   Due Epoch   Type
  me    1          7          4        1       3            5      Any
  me    1          7          4        2       3            5      Any
  me    1          7          4        3       4            6      Any
  me    1          8          4        1       4            6      Any
  me    1          8          4        2       4            6      Any
  me    1          8          4        3       5            7      Any

#   From Node   To Node   Start Epoch   ID   Size (MB)   Due Epoch   Type
  me    1          6          5        1       2            6      Any
  me    1          6          5        2       3            7      Any
  me    1          7          5        1       2            6      Any
  me    1          7          5        2       1            7      Any
  me    1          8          5        1       3            6      Any
  me    1          8          5        2       3            7      Any

#   From Node   To Node   Start Epoch   ID   Size (MB)   Due Epoch   Type
  me    1          6          6        1       2            7      Any
  me    1          7          6        1       1            7      Any
  me    1          8          6        1       2            7      Any

###############################################################################
# messages sent from node 2
###############################################################################

#   From Node   To Node   Start Epoch   ID   Size (MB)   Due Epoch   Type
  me    2          6          1        1       3            2      Any
  me    2          7          1        1       2            4      Any
  me    2          8          1        1       1            4      Any
  me    2          8          1        2       2            5      Any

#   From Node   To Node   Start Epoch   ID   Size (MB)   Due Epoch   Type
  me    2          6          2        1       2            3      Any
  me    2          6          2        2       4            3      Any
  me    2          6          2        3       3            3      Any
```

```
  me    2          7          2        1       3            4      Any
  me    2          7          2        2       4            4      Any
  me    2          7          2        3       4            3      Any
  me    2          8          2        1       2            3      Any

#   From Node   To Node   Start Epoch   ID   Size (MB)   Due Epoch   Type
  me    2          6          3        1       3            4      Any
  me    2          6          3        2       3            4      Any
  me    2          6          3        3       5            7      Any
  me    2          7          3        1       5            4      Any
  me    2          7          3        2       5            4      Any
  me    2          7          3        2       1            5      Any
  me    2          8          3        1       3            5      Any
  me    2          8          3        2       3            6      Any

#   From Node   To Node   Start Epoch   ID   Size (MB)   Due Epoch   Type
  me    2          6          4        1       6            5      Any
  me    2          7          4        1       5            5      Any
  me    2          7          4        2       5            6      Any
  me    2          7          4        3       3            7      Any

#   From Node   To Node   Start Epoch   ID   Size (MB)   Due Epoch   Type
  me    2          6          5        1       4            6      Any
  me    2          7          5        1       4            6      Any
  me    2          8          5        1       3            6      Any
  me    2          8          5        2       2            7      Any

#   From Node   To Node   Start Epoch   ID   Size (MB)   Due Epoch   Type
  me    2          6          6        1       2            7      Any
  me    2          7          6        1       2            7      Any
  me    2          8          6        1       2            7      Any

################################################################################
# messages sent from node 3
################################################################################

#   From Node   To Node   Start Epoch   ID   Size (MB)   Due Epoch   Type
  me    3          6          1        1       3            2      Any
  me    3          6          1        2       1            3      Any
  me    3          7          1        1       3            2      Any
  me    3          7          1        2       4            2      Any
  me    3          8          1        1       3            2      Any
  me    3          8          1        2       4            4      Any

#   From Node   To Node   Start Epoch   ID   Size (MB)   Due Epoch   Type
  me    3          6          2        1       4            3      Any
  me    3          6          2        2       3            3      Any
  me    3          7          2        1       2            3      Any
  me    3          8          2        1       2            4      Any
  me    3          8          2        2       4            5      Any
  me    3          8          2        3       2            3      Any

#   From Node   To Node   Start Epoch   ID   Size (MB)   Due Epoch   Type
  me    3          6          3        1       6            4      Any
  me    3          6          3        2       3            4      Any
  me    3          8          3        1       4            4      Any
  me    3          8          3        2       5            5      Any
```

```
#   From Node   To Node   Start Epoch   ID   Size (MB)   Due Epoch   Type
  me    3          6           4        1       2            5      Any
  me    3          6           4        2       3            5      Any
  me    3          6           4        3       2            6      Any
  me    3          7           4        1       5            6      Any
  me    3          7           4        2       2            5      Any
  me    3          7           4        3       4            7      Any
  me    3          8           4        1       2            5      Any
  me    3          8           4        2       3            6      Any
  me    3          8           4        3       5            7      Any

#   From Node   To Node   Start Epoch   ID   Size (MB)   Due Epoch   Type
  me    3          6           5        1       2            6      Any
  me    3          6           5        2       2            7      Any
  me    3          7           5        1       3            6      Any
  me    3          7           5        2       2            7      Any
  me    3          8           5        1       2            6      Any
  me    3          8           5        2       1            7      Any

#   From Node   To Node   Start Epoch   ID   Size (MB)   Due Epoch   Type
  me    3          6           6        1       1            7      Any
  me    3          7           6        1       1            7      Any
  me    3          8           6        1       1            7      Any
```

A.9 Test File 9

```
################################################################################
#
# PROBLEM DEFINITION (pd)
#
# Note: Epochs begin counting from 0.
#
#      Name                   Number of Epochs   Epoch Size

  pd   s1_p1_c1                      10              1.0

################################################################################
#
# PLATFORMS (pl)
#
# NOTE: Right now, a platform is simply an id number and a name.
#
# TODO: Names sould be delimited by quotes and not require underscores
#       for spaces.
#
#      ID   Name

  pl   1    platform_1
  pl   2    platform_2
  pl   3    platform_3
  pl   4    platform_4
  pl   5    platform_5
  pl   6    platform_6
```

```
  pl   7    satelite_1
  pl   8    satelite_2

################################################################################
#
# ROLES (ro)
#
# NOTE: Right now, a role is simply an id number and a name.
#
#      ID   Name

  ro   1    role_send_1
  ro   2    role_send_2
  ro   3    role_send_3
  ro   4    role_recv_1
  ro   5    role_recv_2
  ro   6    role_recv_3
  ro   7    role_relay_1
  ro   8    role_relay_2

################################################################################
#
# NODES (no)
#
#      ID   Platform   Role   Type         Begin   End

# send nodes
  no   1       1        1     Send           1      10
  no   2       2        2     Send           1      10
  no   3       3        3     Send           1      10

# relay nodes
  no   4       7        7     Relay          1      10
  no   5       8        8     Relay          1      10

# receive nodes
  no   6       4        4     Receive        1      10
  no   7       5        5     Receive        1      10
  no   8       6        6     Receive        1      10

################################################################################
#
# CHANNELS (ch)
#
# Note: Channels are bi-directional between nodes.
#
#      ID   From   To   Type   Bandwidth   Threshold   Cost   Begin   End

# channels out of node 1
  ch   1     1      8    Any       5           80        1      1     10
  ch   2     1      4    Any       10          80        1      1     10
  ch   3     1      7    Any       5           80        1      1     10
  ch   4     1      5    Any       10          80        1      1     10

# channels out of node 2
  ch   5     2      8    Any       5           80        1      1     10
  ch   6     2      4    Any       10          80        1      1     10
```

```
  ch   7     2      5     Any       10          80        1      1     10
  ch   8     2      6     Any       5           80        1      1     10

# channels out of node 3
  ch   9     3      4     Any       10          80        1      1     10
  ch   10    3      5     Any       5           80        1      1     10
  ch   11    3      6     Any       5           80        1      1     10

# channels out of node 4
  ch   12    4      5     Any       25          80        1      1     10
  ch   13    4      6     Any       10          80        1      1     10
  ch   14    4      7     Any       10          80        1      1     10
  ch   15    4      8     Any       10          80        1      1     10

# channels out of node 5
  ch   16    5      6     Any       5           80        1      1     10
  ch   17    5      7     Any       5           80        1      1     10
  ch   18    5      8     Any       10          80        1      1     10

###############################################################################
#
# MESSAGES (me)
#
#   From Node   To Node   Start Epoch   ID   Size (MB)   Due Epoch   Type

###############################################################################
# messages sent from node 1
###############################################################################

#   From Node   To Node   Start Epoch   ID   Size (MB)   Due Epoch   Type
  me    1          6          1        1       2            2      Any
  me    1          7          1        1       3            3      Any
  me    1          8          1        1       2            2      Any

#   From Node   To Node   Start Epoch   ID   Size (MB)   Due Epoch   Type
  me    1          6          2        1       2            3      Any
  me    1          6          2        2       3            4      Any
  me    1          7          2        1       2            3      Any
  me    1          7          2        2       3            4      Any
  me    1          7          2        3       1            5      Any
  me    1          8          2        1       2            3      Any
  me    1          8          2        2       3            4      Any

#   From Node   To Node   Start Epoch   ID   Size (MB)   Due Epoch   Type
  me    1          6          3        1       4            6      Any
  me    1          6          3        2       5            7      Any
  me    1          6          3        3       4            5      Any
  me    1          7          3        1       3            4      Any
  me    1          7          3        2       2            5      Any
  me    1          8          3        1       3            4      Any
  me    1          8          3        2       3            4      Any
  me    1          8          3        3       4            5      Any

#   From Node   To Node   Start Epoch   ID   Size (MB)   Due Epoch   Type
  me    1          7          4        1       3            5      Any
  me    1          7          4        2       3            5      Any
  me    1          7          4        3       4            6      Any
```

```
  me    1          8          4        1       4            6      Any
  me    1          8          4        2       4            6      Any
  me    1          8          4        3       5            7      Any

#   From Node   To Node   Start Epoch   ID   Size (MB)   Due Epoch   Type
  me    1          6          5        1       2            6      Any
  me    1          6          5        2       3            7      Any
  me    1          7          5        1       2            6      Any
  me    1          7          5        2       1            7      Any
  me    1          8          5        1       3            6      Any
  me    1          8          5        2       3            7      Any

#   From Node   To Node   Start Epoch   ID   Size (MB)   Due Epoch   Type
  me    1          6          6        1       2            7      Any
  me    1          7          6        1       1            7      Any
  me    1          8          6        1       2            7      Any

###############################################################################
# messages sent from node 2
###############################################################################

#   From Node   To Node   Start Epoch   ID   Size (MB)   Due Epoch   Type
  me    2          6          1        1       3            2      Any
  me    2          7          1        1       2            4      Any
  me    2          8          1        1       1            4      Any
  me    2          8          1        2       2            5      Any

#   From Node   To Node   Start Epoch   ID   Size (MB)   Due Epoch   Type
  me    2          6          2        1       2            3      Any
  me    2          6          2        2       4            3      Any
  me    2          6          2        3       3            3      Any
  me    2          7          2        1       3            4      Any
  me    2          7          2        2       4            4      Any
  me    2          7          2        3       4            3      Any
  me    2          8          2        1       2            3      Any

#   From Node   To Node   Start Epoch   ID   Size (MB)   Due Epoch   Type
  me    2          6          3        1       3            4      Any
  me    2          6          3        2       3            4      Any
  me    2          6          3        3       5            7      Any
  me    2          7          3        1       5            4      Any
  me    2          7          3        2       5            4      Any
  me    2          7          3        2       1            5      Any
  me    2          8          3        1       3            5      Any
  me    2          8          3        2       3            6      Any

#   From Node   To Node   Start Epoch   ID   Size (MB)   Due Epoch   Type
  me    2          6          4        1       6            5      Any
  me    2          7          4        1       5            5      Any
  me    2          7          1        2       5            6      Any
  me    2          7          4        3       3            7      Any

#   From Node   To Node   Start Epoch   ID   Size (MB)   Due Epoch   Type
  me    2          6          5        1       4            6      Any
  me    2          7          5        1       4            6      Any
  me    2          8          5        1       3            6      Any
  me    2          8          5        2       2            7      Any
```

```
#   From Node   To Node   Start Epoch   ID   Size (MB)   Due Epoch   Type
  me    2          6           6        1       2            7      Any
  me    2          7           6        1       2            7      Any
  me    2          8           6        1       2            7      Any

###############################################################################
# messages sent from node 3
###############################################################################

#   From Node   To Node   Start Epoch   ID   Size (MB)   Due Epoch   Type
  me    3          6           1        1       3            2      Any
  me    3          6           1        2       1            3      Any
  me    3          7           1        1       3            2      Any
  me    3          7           1        2       4            2      Any
  me    3          8           1        1       3            2      Any
  me    3          8           1        2       4            4      Any

#   From Node   To Node   Start Epoch   ID   Size (MB)   Due Epoch   Type
  me    3          6           2        1       4            3      Any
  me    3          6           2        2       3            3      Any
  me    3          7           2        1       2            3      Any
  me    3          8           2        1       2            4      Any
  me    3          8           2        2       4            5      Any
  me    3          8           2        3       2            3      Any

#   From Node   To Node   Start Epoch   ID   Size (MB)   Due Epoch   Type
  me    3          6           3        1       6            4      Any
  me    3          6           3        2       3            4      Any
  me    3          8           3        1       4            4      Any
  me    3          8           3        2       5            5      Any

#   From Node   To Node   Start Epoch   ID   Size (MB)   Due Epoch   Type
  me    3          6           4        1       2            5      Any
  me    3          6           4        2       3            5      Any
  me    3          6           4        3       2            6      Any
  me    3          7           4        1       5            6      Any
  me    3          7           4        2       2            5      Any
  me    3          7           4        3       4            7      Any
  me    3          8           4        1       2            5      Any
  me    3          8           4        2       3            6      Any
  me    3          8           4        3       5            7      Any

#   From Node   To Node   Start Epoch   ID   Size (MB)   Due Epoch   Type
  me    3          6           5        1       2            6      Any
  me    3          6           5        2       2            7      Any
  me    3          7           5        1       3            6      Any
  me    3          7           5        2       2            7      Any
  me    3          8           5        1       2            6      Any
  me    3          8           5        2       1            7      Any

#   From Node   To Node   Start Epoch   ID   Size (MB)   Due Epoch   Type
  me    3          6           6        1       1            7      Any
  me    3          7           6        1       1            7      Any
  me    3          8           6        1       1            7      Any
```

A.10 Test File 10

```
################################################################################
#
# PROBLEM DEFINITION (pd)
#
# Note: Epochs begin counting from 0.
#
#      Name                   Number of Epochs   Epoch Size

  pd   s1_p1_c1                     10              1.0

################################################################################
#
# PLATFORMS (pl)
#
# NOTE: Right now, a platform is simply an id number and a name.
#
# TODO: Names sould be delimited by quotes and not require underscores
#       for spaces.
#
#      ID   Name

  pl   1    platform_1
  pl   2    platform_2
  pl   3    platform_3
  pl   4    platform_4
  pl   5    platform_5
  pl   6    platform_6
  pl   7    satelite_1
  pl   8    satelite_2

################################################################################
#
# ROLES (ro)
#
# NOTE: Right now, a role is simply an id number and a name.
#
#      ID   Name

  ro   1    role_send_1
  ro   2    role_send_2
  ro   3    role_send_3
  ro   4    role_recv_1
  ro   5    role_recv_2
  ro   6    role_recv_3
  ro   7    role_relay_1
  ro   8    role_relay_2

################################################################################
#
# NODES (no)
#
#      ID   Platform   Role   Type         Begin   End

# send nodes
```

```
  no   1       1        1     Send          1      10
  no   2       2        2     Send          1      10
  no   3       3        3     Send          1      10

# relay nodes
  no   4       7        7     Relay         1      10
  no   5       8        8     Relay         1      10

# receive nodes
  no   6       4        4     Receive       1      10
  no   7       5        5     Receive       1      10
  no   8       6        6     Receive       1      10

###############################################################################
#
# CHANNELS (ch)
#
# Note: Channels are bi-directional between nodes.
#
#      ID   From   To   Type   Bandwidth   Threshold   Cost   Begin   End

# channels out of node 1
  ch   1     1      8    Any       5           80        1      1     10
  ch   2     1      4    Any       10          80        1      1     10
  ch   3     1      7    Any       5           80        1      1     10
  ch   4     1      5    Any       10          80        1      1     10

# channels out of node 2
  ch   5     2      8    Any       5           80        1      1     10
  ch   6     2      4    Any       10          80        1      1     10
  ch   7     2      5    Any       10          80        1      1     10
  ch   8     2      6    Any       5           80        1      1     10

# channels out of node 3
  ch   9     3      4    Any       10          80        1      1     10
  ch   10    3      5    Any       5           80        1      1     10
  ch   11    3      6    Any       5           80        1      1     10

# channels out of node 4
  ch   12    4      5    Any       25          80        1      1     10
  ch   13    4      6    Any       10          80        1      1     10
  ch   14    4      7    Any       10          80        1      1     10
  ch   15    4      8    Any       10          80        1      1     10

# channels out of node 5
  ch   16    5      6    Any       5           80        1      1     10
  ch   17    5      7    Any       5           80        1      1     10
  ch   18    5      8    Any       10          80        1      1     10

###############################################################################
#
# MESSAGES (me)
#
#   From Node   To Node   Start Epoch   ID   Size (MB)   Due Epoch   Type

###############################################################################
# messages sent from node 1
```

```
################################################################################

#   From Node   To Node   Start Epoch   ID   Size (MB)   Due Epoch   Type
  me    1          6          1        1       2            2      Any
  me    1          7          1        1       3            3      Any
  me    1          8          1        1       2            2      Any

#   From Node   To Node   Start Epoch   ID   Size (MB)   Due Epoch   Type
  me    1          6          2        1       2            3      Any
  me    1          6          2        2       3            4      Any
  me    1          7          2        1       2            3      Any
  me    1          7          2        2       3            4      Any
  me    1          7          2        3       1            5      Any
  me    1          8          2        1       2            3      Any
  me    1          8          2        2       3            4      Any

#   From Node   To Node   Start Epoch   ID   Size (MB)   Due Epoch   Type
  me    1          6          3        1       4            6      Any
  me    1          6          3        2       5            7      Any
  me    1          6          3        3       4            5      Any
  me    1          7          3        1       3            4      Any
  me    1          7          3        2       2            5      Any
  me    1          8          3        1       3            4      Any
  me    1          8          3        2       3            4      Any
  me    1          8          3        3       4            5      Any

#   From Node   To Node   Start Epoch   ID   Size (MB)   Due Epoch   Type
  me    1          7          4        1       3            5      Any
  me    1          7          4        2       3            5      Any
  me    1          7          4        3       4            6      Any
  me    1          8          4        1       4            6      Any
  me    1          8          4        2       4            6      Any
  me    1          8          4        3       5            7      Any

#   From Node   To Node   Start Epoch   ID   Size (MB)   Due Epoch   Type
  me    1          6          5        1       2            6      Any
  me    1          6          5        2       3            7      Any
  me    1          7          5        1       2            6      Any
  me    1          7          5        2       1            7      Any
  me    1          8          5        1       3            6      Any
  me    1          8          5        2       3            7      Any

#   From Node   To Node   Start Epoch   ID   Size (MB)   Due Epoch   Type
  me    1          6          6        1       2            7      Any
  me    1          7          6        1       1            7      Any
  me    1          8          6        1       2            7      Any

################################################################################
# messages sent from node 2
################################################################################

#   From Node   To Node   Start Epoch   ID   Size (MB)   Due Epoch   Type
  me    2          6          1        1       3            2      Any
  me    2          7          1        1       2            4      Any
  me    2          8          1        1       1            4      Any
  me    2          8          1        2       2            5      Any
```

```
#   From Node   To Node   Start Epoch   ID   Size (MB)   Due Epoch   Type
  me    2          6           2        1       2            3      Any
  me    2          6           2        2       4            3      Any
  me    2          6           2        3       3            3      Any
  me    2          7           2        1       3            4      Any
  me    2          7           2        2       4            4      Any
  me    2          7           2        3       4            3      Any
  me    2          8           2        1       2            3      Any

#   From Node   To Node   Start Epoch   ID   Size (MB)   Due Epoch   Type
  me    2          6           3        1       3            4      Any
  me    2          6           3        2       3            4      Any
  me    2          6           3        3       5            7      Any
  me    2          7           3        1       5            4      Any
  me    2          7           3        2       5            4      Any
  me    2          7           3        2       1            5      Any
  me    2          8           3        1       3            5      Any
  me    2          8           3        2       3            6      Any

#   From Node   To Node   Start Epoch   ID   Size (MB)   Due Epoch   Type
  me    2          6           4        1       6            5      Any
  me    2          7           4        1       5            5      Any
  me    2          7           4        2       5            6      Any
  me    2          7           4        3       3            7      Any

#   From Node   To Node   Start Epoch   ID   Size (MB)   Due Epoch   Type
  me    2          6           5        1       4            6      Any
  me    2          7           5        1       4            6      Any
  me    2          8           5        1       3            6      Any
  me    2          8           5        2       2            7      Any

#   From Node   To Node   Start Epoch   ID   Size (MB)   Due Epoch   Type
  me    2          6           6        1       2            7      Any
  me    2          7           6        1       2            7      Any
  me    2          8           6        1       2            7      Any

################################################################################
# messages sent from node 3
################################################################################

#   From Node   To Node   Start Epoch   ID   Size (MB)   Due Epoch   Type
  me    3          6           1        1       6            2      Any
  me    3          6           1        2       1            3      Any
  me    3          7           1        1       3            2      Any
  me    3          7           1        2       4            2      Any
  me    3          8           1        1       3            2      Any
  me    3          8           1        2       4            4      Any

#   From Node   To Node   Start Epoch   ID   Size (MB)   Due Epoch   Type
  me    3          6           2        1       4            3      Any
  me    3          6           2        2       3            3      Any
  me    3          7           2        1       2            3      Any
  me    3          8           2        1       2            4      Any
  me    3          8           2        2       4            5      Any
  me    3          8           2        3       2            3      Any

#   From Node   To Node   Start Epoch   ID   Size (MB)   Due Epoch   Type
```

```
  me    3          6          3        1       6            4      Any
  me    3          6          3        2       3            4      Any
  me    3          8          3        1       4            4      Any
  me    3          8          3        2       5            5      Any

#   From Node   To Node   Start Epoch   ID   Size (MB)   Due Epoch   Type
  me    3          6          4        1       2            5      Any
  me    3          6          4        2       3            5      Any
  me    3          6          4        3       2            6      Any
  me    3          7          4        1       5            6      Any
  me    3          7          4        2       2            5      Any
  me    3          7          4        3       4            7      Any
  me    3          8          4        1       2            5      Any
  me    3          8          4        2       3            6      Any
  me    3          8          4        3       5            7      Any

#   From Node   To Node   Start Epoch   ID   Size (MB)   Due Epoch   Type
  me    3          6          5        1       2            6      Any
  me    3          6          5        2       2            7      Any
  me    3          7          5        1       3            6      Any
  me    3          7          5        2       2            7      Any
  me    3          8          5        1       2            6      Any
  me    3          8          5        2       1            7      Any

#   From Node   To Node   Start Epoch   ID   Size (MB)   Due Epoch   Type
  me    3          6          6        1       1            7      Any
  me    3          7          6        1       1            7      Any
  me    3          8          6        1       1            7      Any
```

Vita

Jon was born on May 19, 1978 in Richmond, Virginia. He earned a B.A. degree in Mathematics in 2001 from Roanoke College, Salem Virginia. From there, he went on to persue a graduate degree in Computer Science from Virginia Tech. While there, his research focused on communication networks and graph theory. This thesis completes his M.S. degree in Computer Science from Virginia Tech.