4-2008

A Proof of Concept for Oppnets and Its Resource Utilization Techniques with QOS Constraints

Zill-E-Huma Kamal
Western Michigan University

A PROOF OF CONCEPT FOR OPPNETS AND ITS RESOURCE UTILIZATION TECHNIQUES WITH QOS CONSTRAINTS

by

Zill-E-Huma Kamal

A Dissertation
Submitted to the
Faculty of The Graduate College
in partial fulfillment of the
requirements for the
Degree of Doctor of Philosophy
Department of Computer Science
Dr. Ajay Gupta, Advisor

Western Michigan University
Kalamazoo, Michigan
April 2008

A PROOF OF CONCEPT FOR OPPNETS AND ITS RESOURCE UTILIZATION TECHNIQUES WITH QoS CONSTRAINTS

Zill-E-Huma Kamal, Ph.D.

Western Michigan University, 2008

The increased number of embedded devices/systems in our environment and the evolution of the internet as a service oriented network connote two things: the demand for ubiquitous computing, and, the abstraction of users as consumers and applications as services, as in Service Oriented Computing (SOC) and Web Service domains.

This Thesis studies Class 2 Opportunistic Networks, Oppnets—in short. Oppnets propagate dynamic interconnection of heterogeneous devices/networks/systems and the integration of their resources or services–such as computation, communication, sensing, actuation, and storage – *irrespective* of the discrepancies in hardware, software, protocols, or standards employed, to enable ubiquitous computing. The novelty of Oppnets lies in its ability to grow into a larger network and leverage resources of the new heterogenous devices/networks/systems as though they were part of the initial network.

The first contribution of this research is the design and implementation of a small-scale Oppnet, called MicroOppnet, which not only acts as a proof-of-concept for Oppnets but can also be extended to be a testbed for experimentation and pilot implementation of Oppnet architectures and their components.

Since the Oppnet idea is still in its infancy, there are numerous challenges confronting Oppnets, amongst them is resource utilization. We present a novel Service Location and Planning (SLP) mechanism that enables resource utilization in Oppnets.

The second contribution of this Thesis, is the definition and implementation of the

novel SLP problem as a mathematical model that can be solved *optimally* for small-scale networks. We also solve SLP problem for large networks using Lagrangean Relaxation.

The SLP mechanism meets consumers' requests, by installing the requested service on a node, that not only minimizes service installation costs, but also promotes service federation (i.e. multiple services installed on a node), and abides by consumer-defined quality of service (QoS) and realistic network parameters. This realistic modeling accounts for consumer-defined QoS constraints of throughput and delay, the underlying network link layer bandwidth capacities, and the important factor of cost to the provider.

In this Thesis, we show feasibility of Oppnets, with the design and implementation of MicroOppnet and discuss the SLP mechanism and its implementation and application to Oppnets.

UMI Number: 3303468

ACKNOWLEDGMENTS

This dissertation has been like a journey, whose map was given to me by my parents, Mohammed Anwar and Naheed Kamal. Their inspiration and support have been instrumental in my achievements. I dedicate this dissertation to them.

Despite having a map, I could not have reached this final destination without the supervision and guidance of my "GPS System", my advisor, Dr. Ajay Gupta. The dedication he has shown and efforts he has made to help me succeed are profoundly moving. I would also like to thank all my committee members, Dr. Ala Al-Fuqaha, Dr. Leszek Lilien, Dr. Ikhlas Abdel-Qader and Dr. Matt Mutka and professors, such as Dr. Dionysios Kountanis, Dr. Nelson and Dr. Kaminiski, for their consideration, assistance and support. During this journey I have also had the pleasure of meeting and working with phenomenal peers like Vijay Bhuse and Osama Awwad.

Furthermore, without a companion, embarking on a journey can be daunting. I found my companion for this journey and my life in my husband, Mohammad Ali Salahuddin, without his encouragement and support this journey would have been impossible. My brother, Muneeb Kamal, is as always, my partner in fun. The support of my in-laws, Hashim Ali Khan and Shakera Hashim, was also vital in making this dissertation possible.

I am forever indebted to all those mentioned and many more for making this journey a dream come true for me.

Zill-E-Huma Kamal

TABLE OF CONTENTS

Table of Contents—continued

CHAPTER

CHAPTER

LIST OF TABLES

LIST OF FIGURES

CHAPTER 1

INTRODUCTION

This chapter discusses Pervasive Service Oriented Computing and Class 2 Opportunistic Networks as an example of a Pervasive Service Oriented Computing network. We briefly discuss challenges in this paradigm and give motivation for the research in this Thesis.

1.1 Motivation

The Internet has evolved from a simple, restricted network allowing communication between fixed points to a mesh of global internetworking technology. With this evolution various computing paradigms have sprouted up, ranging from Mobile Computing to Wireless Sensor Networking to Pervasive Computing to the emerging and still evolving Web Services and Service Oriented Computing.

A closer look at any of these paradigms and technologies will give insight into the direction of research for that paradigm and its demands. For example, Mobile Computing. Its goal is simple, that is, to remain connected to the global Internet untethered and still have the ability to perform the various tasks and operate applications as though connected to the Internet traditionally (physically). The demand from Mobile Computing is simple: seamless ubiquitous connectivity. The research in that paradigm is continuously trying to meet this demand via research into optimal coverage schemes, connectivity speeds, mobile wireless internet standards, etc.

Similarly, consider Pervasive Computing. Its goal is to "converge computers, communication, consumer electronics, content and services, such that devices interact over two underlying layers, namely, service and standards" [1]. The service layer establishes infrastructure for computing, communication, content, access, etc. The standards layer allows information and application exchange (e.g. standards include, Java, XML, HTML, etc.) [1]. The demand from Pervasive Computing is simple: seamless ubiquitous computing and communications. The research for this paradigm is continuously trying to achieve the above mentioned goals.

Let's consider the evolving Service Oriented Computing (SOC), where services provide a higher-level abstraction to traditional applications and users are considered as consumers of the services. Consumers lookup, via registries, providers/producers of these services. The goal again is to meet the demand for ubiquitous computing through a higher-level abstraction of applications as services.

Therefore, it can be seen that the current demand from this evolving global internetworking technology is the same – seamless, untethered, ubiquitous computing and communication, which is continuously being explored and marketed.

Meeting this demand is in essence a two-fold approach: first, developing a ubiquitous environment, and second, developing a mechanism for mapping consumers to providers/producers.

Bearing in mind this demand and the two-fold approach to meet this demand, this research discusses a new computing paradigm that can provide the ubiquitous environment, and a service location-planning approach that defines and solves the problem of mapping providers/producers with consumers.

The new computing paradigm discussed in this Thesis, termed Opportunistic Networks (Oppnet), is a seamless integration of heterogeneous devices, networks, and systems operating on or using disjoint communication media into a *new network*–an Opportunistic Network (Oppnet). However, an Oppnet is not only an integration of hardware; it is also the integration of the resources of the independent devices, networks, and systems, this enables the Oppnet to "grow" into a larger network with every new device, network, and system that is integrated into the Oppnet. This integration of resources can then be used to collaboratively execute tasks that may not have been feasible for any independent device, network, and system in the Oppnet.

A detailed discussion of Opportunistic Networks (Oppnets) will be presented shortly.

1.2 Thesis Statement

The contribution of this research is to present the design and implementation of a small-scale Oppnet that will not only act as a proof of concept but also as a simple Oppnet prototype. Using this small-scale prototype to establish feasibility of an Oppnet, this research tackles the second aspect, namely, mapping producers to consumers. This is attained by defining a Service Location-Planning (SLP) Problem that is solved optimally for small-scale networks and solved approximately with near-optimal results for large-scale networks.

1.3 Thesis Outline

The rest of the Thesis is organized as follows. Chapter 2 first presents and discusses Oppnets: their categories, basic ideas and operations, and control flow. It is followed by a discussion of the service-level abstraction of applications and the Service Location-Planning (SLP) problem. Chapter 3 first gives the literature review of other technologies and standards for developing the ubiquitous environments we discussed. It is followed by a review of various similar optimization problems that occur in different fields and their comparison with Service Location-Planning (SLP) Problem researched in this Thesis. Chapter 4 presents the design and implementation of the small-scale Oppnet, called MicroOppnet, built during this research. The Thesis continues with presentation and definition of the Service Location-Planning (SLP) Problem, an Integer Linear Model (Integer Linear Programming Problem) to solve the SLP optimally for small-scale networks in chapter 5. In chapter 6 first the Lagrangean Relaxation approximation technique is discussed and then used to solve SLP for large-scale networks. Chapter 7 concludes with the thrust of the research of this Thesis and presents a summary of the contributions and future work.

CHAPTER 2

BACKGROUND

The first mention of the idea of Opportunistic Sensor Networks was made by Lilien [5]. Later Lilien et al. further investigated, researched, and developed Opportunistic Networks in [3, 4, 6]—to mention a few. Below is a reproduction of relevant text to discuss categorization of Opportunistic Networks, definition of Class 2 Opportunistic Networks (Oppnets), their basic ideas, operations, control flow, etc. [3, 4, 6, 25].

Following this detailed discussion of Oppnets, the Service Location-Planning (SLP) problem is discussed.

2.1 Categories of Opportunistic Networks

Over the years, numerous technologies enabling pervasive computing have been proposed, researched and experimented with, to meet demands for ubiquitous communication, computation, data processing, storage, etc. Various technologies have emerged, such as P2P computing networks, grid networks, mesh networks, ambient networks, and, most recently, opportunistic networks.

In *Class 1 Opportunistic Networks* [3], opportunism is quite restricted, usually limited to opportunistic connectivity that is, establishing communications when devices are within each other's range [2]. In contrast, in the new paradigm and technology called

Class 2 Opportunistic Networks, enables not only opportunistic communication but also an opportunistic growth of networks and opportunistic use of resources gained by this growth [4].

Effectively, Oppnets leverage their capabilities by exploiting the wealth of resources available on all kinds of pervasive devices that are within their reach—crossing communication, hardware and software barriers.

Class 1 Opportunistic Networks extend communication capabilities while Class 2 in addition provides services that they are able to discover (also via true discovery, not just a directory lookup).

Opportunistic data dissemination techniques [15–17] might be considered *"Class 1.5" Opportunistic Networks.*

2.2 Class 2 Opportunistic Networks (Oppnets)

Each Class 2 Opportunistic Network grows from a *seed Oppnet*, or simply a *seed*, which is a set of nodes employed together at the time of the initial Oppnet deployment. A seed can be wireless—with nodes communicating via radio channels, and ad hoc—with nodes not carefully pre-positioned but, for instance, thrown out of a plane or a moving car in the deployment area. Seed nodes for demanding applications, e.g. for emergency operations, could be quite powerful, such as powerful mobile communication/computing/sensing hosts mounted on heavy all-terrain trucks or amphibious vehicles, or in parachute-dropped containers. For less demanding applications, seed nodes could even be arbitrarily lightweight.

After the seed self configures and becomes operational, its first task is to detect a set of "reserve" nodes and "foreign" entities—devices, node clusters, networks, and other systems—which it deems useful. *Reserve nodes* are those that are already pre-configured with Oppnet-enabled code, whereas *foreign* nodes are those that do not have any a priori information about oppnets protocols. Detected entities are candidates for becoming *helpers* for the Oppnet. Each such *candidate helper* (*candidate*) has a potential to provide Oppnet with some communication, computing, sensing, or other capabilities or resources.

Candidates are evaluated by the Oppnet, and those that can help in achieving the goals of the Oppnet are ordered or invited to join the Oppnet (see Section 2.3.3 for discussion of when candidates are invited or ordered to join Oppnet). Invited candidates can either accept or refuse the invitation.

It is important to note that by admitting helpers an Oppnet can leverage all kinds of resources it needs that are available in its environment. This is Oppnet growth, and is a mechanism to obtain a lot of help at a very low cost. This Oppnet growth mechanism distinguishes Oppnet from other Opportunistic Networks and other pervasive computing technologies.

Oppnets can be envisioned in numerous scenarios from the mundane to military applications. Oppnets are also a fit for Emergency Preparedness and Response (EPR) applications that are currently in the limelight due to the tragic events of September 11^{th}, Hurricane Katrina, California Wildfires, and alike. In the sections that follow Oppnets are discussed in terms of an EPR application scenario, which is one sample application area.

2.3 Basic Oppnet Ideas and Operations

This section presents a detailed discussion of basic Oppnet ideas, operations and terminology and heavily borrows from our earlier preliminary work in [3, 4, 6].

2.3.1 Seed Oppnets and Oppnet helpers

Seed Oppnets: Each Oppnet starts as a seed Oppnet—a set of nodes employed together at the time of the initial network deployment, as illustrated in Fig. 1. The seed is predesigned, and can be viewed as a network in its own right. It might be very small, in the extreme consisting of a single node.

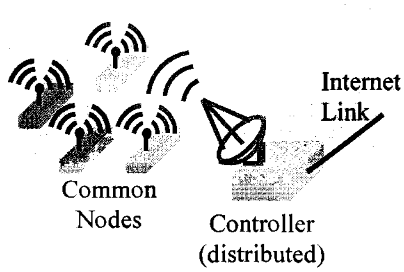


Figure 1. Seed Oppnet ([3], [4])

A subset of seed nodes constitutes a distributed Control Center (CC). CC can grow admitting other nodes, and can shrink expelling any of its nodes. Admitted nodes are called *helpers*. We can have both regular helpers and lightweight helpers or lites (such as a smoke detector). *Lites* are Oppnet-enabled, that is equipped with inexpensive, simple means of standard Oppnet communications. In this way, even lites can be triggered to operate in the Oppnet mode when needed and commanded to do so by a CC or regular helpers.

Summarizing, a node belongs to one of the four categories: (i) CC nodes; (ii) "seed nodes," which really are the seed nodes that are not CC nodes; (iii) "helpers," which really are the regular helpers with reasonable computing and communication capabilities that are not lites; and (iv) lites.

Potential helpers and their discovery: In general, the set of potential helpers for Oppnets is very broad, including communication, computing and sensor systems, both wired and wireless, both free-standing and embedded. As pervasive computing progresses, the candidate pool will continue increasing dramatically: in infrastructures, buildings, vehicles, appliances, etc.

More densely populated areas will have, in general, a denser coverage by potential helpers. Thus, it will be easier to leverage capabilities of an Oppnet in such areas. This is a desirable property in, for example, disaster recovery applications: more resources become available in areas with a possibility of more human victims and more property damage.

Before a seed Oppnet can grow, it must *discover* its own set of potential helpers available to it. In addition to a mere *lookup* of a previously prepared information (e.g., a directory), which is often referred to as "discovery," we mean also much more challenging true discovery. True discovery could involve an Oppnet node scanning the spectrum for signals or beacons, and collecting enough information to contact their senders.

Utilizing helpers: Oppnets can utilize resources of helpers to significantly enhance their capabilities. This has the form of leveraging of all kinds of resources and "skills" (provided by smart or intelligent software) that new helpers bring with them. In this way, Oppnets obtain a lot of help effectively and efficiently (even for free in emergency situations, as discussed later).

Use of helper functionalities can be innovative in at least two ways. First, Oppnets are able to exploit dormant capabilities of their helpers. For instance, even entities with no obvious sensing capabilities can be used for sensing: (a) a desktop can "sense" its user's presence at the keyboard; (b) a smart refrigerator monitoring opening of its door can "sense" presence of potential victims at home in a disaster area. As another example, the water infrastructure sensornet (sensor network) with multisensor capabilities, which is positioned near roads, can be directed to sense vehicular movement, or the lack thereof.

Second, helpers might be used in novel combinations of existing technologies, as in the following scenario [3] and illustrated in Fig. 2.

A seed Oppnet is deployed in a metropolitan area after an earthquake. It finds many potential helpers, and integrates some of them into an expanded Oppnet. One of the nodes of the expanded Oppnet, a surveillance system, "looks" at a public area scene with many objects. The image is passed to an Oppnet node that analyzes it, and recognizes one of the objects as an overturned car. Another node decides that the license plate of the car should be read. As the Oppnet currently includes no image analysis specialist, a helper with such capabilities is found and integrated into the Oppnet. It reads the license plate number. The license plate number is used by another newly integrated helper to check in a vehicle database whether the car is equipped with the vehicular communication system, e.g. OnStar™ [7]. If it is, the appropriate vehicular center facility is contacted, becomes a helper, and obtains a connection with the OnStar™ device in the car.

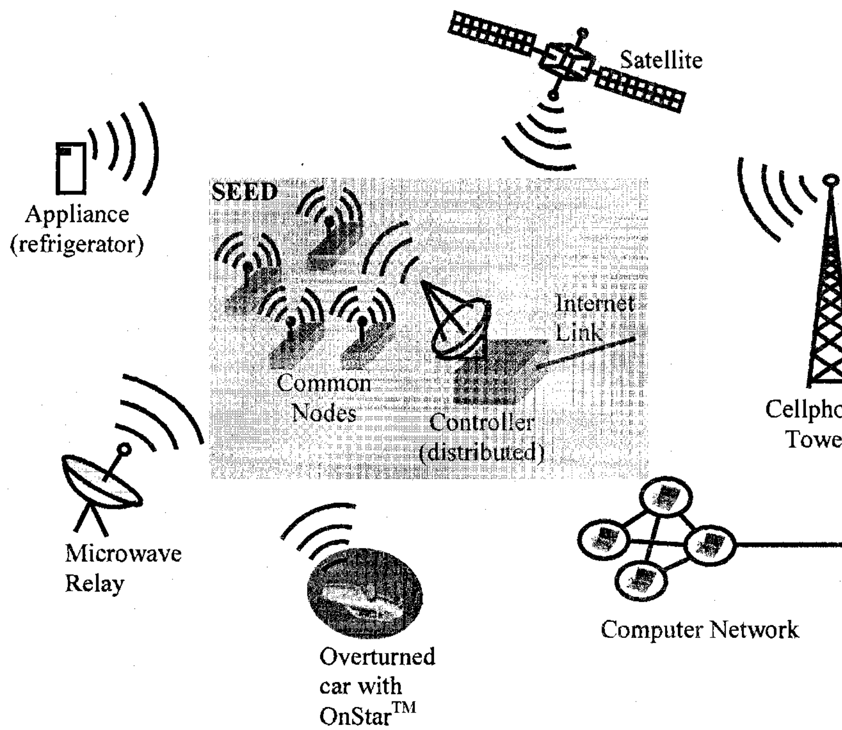


Figure 2. Expanded Oppnet ([3], [4])

The OnStar device in the car becomes a helper and is asked to contact BANs (body area networks) on and within bodies of car occupants. Each BAN available in the car becomes a helper and reports on the vital signs of its owner. The reports from BANs are analyzed by scheduling nodes that schedule the responder teams to ensure that people in the most serious condition are rescued sooner than the ones that can wait for help longer. (Note that with the exception of the BAN link that is just a bit futuristic—its widespread availability could be measured in years not in decades—all other node and helper capabilities used in the scenario are already quite common.)

2.3.2 Growth of seed Oppnet into expanded Oppnet

A seed Oppnet grows into an expanded Oppnet after admitting new helpers. For example, the *expanded Oppnet* in Fig. 2 admitted these helpers: (a) a computer network, contacted via a wired Internet link; (b) a cell phone infrastructure (represented by the cell phone tower), contacted via Oppnet's cell phone peripheral; (c) a satellite, contacted via a direct satellite link; (d) a home area network, contacted via an intelligent appliance (e.g., a refrigerator) with a wireless link; (e) a microwave network, contacted via a microwave relay; (f) BANs of occupants of an overturned car, contacted via OnStar.

Helpers are either invited or ordered to join. In the former case, contacted candidates are free to either join or refuse the invitation. In the latter case for emergency situations, they must accept being conscripted in the spirit of citizens called to arms.

2.3.3 Asking or ordering helpers and Oppnet reserve

Ordering candidate helpers to join may seem controversial, and requires discussion. First, it is obvious that any candidate can be asked to join in any situation.

Second, any candidate can be ordered to join in life-or-death situations. It is an analogy to citizens being required by law to assist with their property (e.g., vehicles) and labor in saving lives or critical resources.

Third, some candidates can always be ordered to become helpers in emergencies. They include many kinds of computing and communication systems serving police, firefighters, the National Guard, etc. Also, the federal/local governments can make some of their systems available upon an order from any EPR (Emergency Preparedness Response) Oppnet.

Once they sign up, they are "trained" for an active duty: facilities assisting Oppnets in their discovery and contacting them are installed on them. For example, standard Oppnet Virtual Machine (OVM) software is installed on them (cf. Appendix A). The "training" makes reservists highly prepared for their Oppnet duties.

Such *Oppnet reserve* is not necessary for the Oppnet paradigm but very helpful for at least two reasons. First, Oppnet *reservists* in an incident area increase the pool of candidates that can be ordered—rather than asked—by an Oppnet to join it. Second, having "trained" reservists (e.g., OVM-equipped ones) significantly simplifies discovery of candidates. Specifically, it facilitates finding by an Oppnet the very first contact in an incident area, which is always most difficult. Once a reservist joins an Oppnet, reservist's own contacts become easy next-wave contacts for the Oppnet.

For EPR applications, we have assumed that at least one reservist survives an incident. With numerous reservists in practically every area of the country—the more reservists the more densely populated is an incident area—we are practically guaranteed that some reservists will survive (and some of the reservists' contacts will survive).

In general, by employing helpers working for free as volunteers or conscripts, Class 2 Opportunistic Networks can be extremely competitive economically in their operation. Full realization of this crucial property requires determining the most appropriate incentives for volunteers and enforcements for conscripts.

To protect critical operations of Oppnets and of helpers joining an Oppnet, Oppnets must obey the following principles:

- Oppnets must not disrupt critical operations of potential helpers. In particular, they must not take over any resources of life-support and life-saving systems.
- For potential helpers running non-critical services, risk evaluation must be performed by an Oppnet before they are asked or ordered to join the Oppnet. This task may be simplified by potential helpers identifying their own risk levels, according to a standard risk level classification.
- Privacy and security of Oppnets and helpers must be assured, especially in the Oppnet growth process.

2.4 Control Flow of Oppnet

The control flow in Oppnets, that is, the basic sequence of Oppnet operations is shown in Fig. 3. Oppnets first deploy a seed Oppnet (cf. Fig. 1), which may be viewed as a pretty typical ad hoc network. It self-configures, and then works to detect "foreign" devices or systems using all kinds of communication media—including wired Internet, WiFi, cell phones, RFIDs, satellites, etc. At this stage, Oppnets start to differ from typical networks.

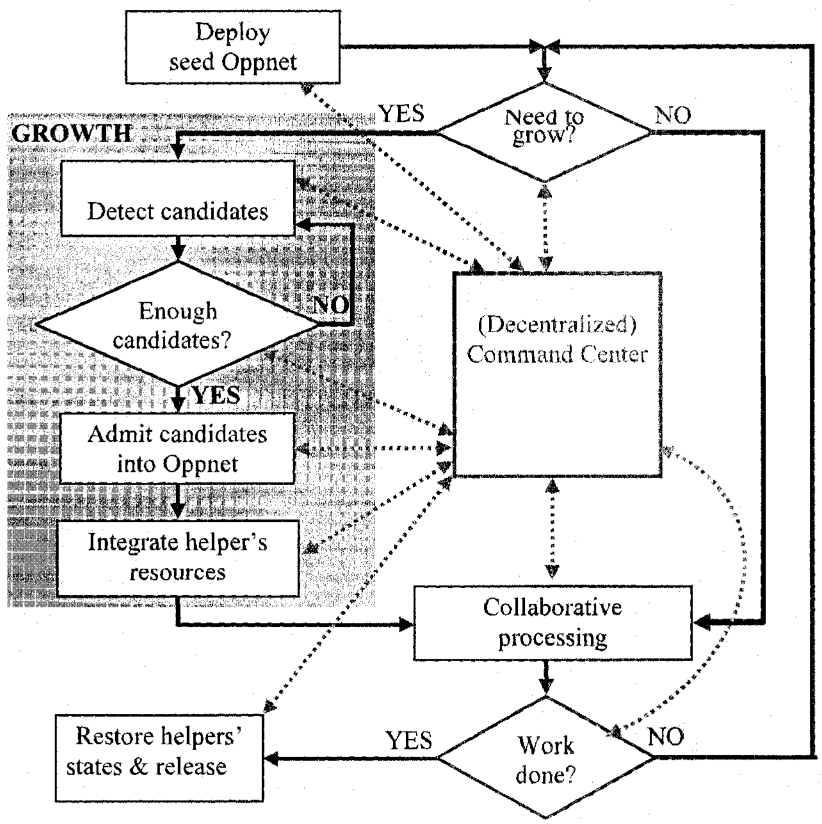


Figure 3. Basic operations of an Oppnet

Detected systems are identified and evaluated for their usefulness and dependability as candidates for joining the Oppnet. The best candidates are invited into the expanded Oppnet. A candidate can accept or reject the invitation (but it might be ordered to join during disaster response operations). Upon accepting the invitation, a candidate is admitted into the Oppnet, becoming its helper. The resources of the helper are integrated with the Oppnet, and Oppnet's tasks can be offloaded to or distributed amongst this and all other helpers (collaborative processing).

A decentralized (distributed) command center—either augmenting human operators or fully autonomous—presides over the operations of the Oppnet throughout its life. If the Oppnet needs more resources to achieve its goal, the process repeats, and once the goal of the Oppnet has been achieved, the helpers are restored and released. It is the goal of the Oppnet to restore helpers' state to the state that the helper was found in before admittance to the Oppnet. This way, Oppnet is minimally intrusive to helpers.

2.5 Open Issues and Challenges in Oppnets

The preceding sections discuss the Oppnet paradigm conceptually, but there are various open issues and challenges in the Oppnet paradigm that must be studied before Oppnets can be realistically implemented. Some of these issues are delineated below [55, 25]:

1. *Optimizing the seed Oppnet infrastructure.* By developing measures and criteria for quantitative specification of Oppnet features such as communication, computation, sensing, storage, energy resources, etc.
2. *Developing methods for detecting helpers with useful resources and facilities.* An integrated solution to detect devices by all diverse technologies, rather than using traditional localization, GPS, or ultra-sound, etc techniques.

2.1 *Designing integrated communication media for Oppnets.* Provide seamless communication across disjoint communication media.

3. *Designing methods for inviting candidate helpers, and methods for controlling helpers.* Develop primitives and protocols for inviting and admitting helpers.

4. *Developing methods/technique for:*

    a) *Deciding which tasks should be "offloaded" by Oppnet to its helpers.*

    b) *Coordinating helper tasks by Oppnets.* Conjure protocols and primitives for collaborating and delegating tasks amongst Oppnet nodes.

5. *Proposing ways of*

    a) *Managing Oppnets.*

    b) *Control of privacy and security problems in Oppnets.* Algorithms for monitoring Oppnet nodes to detect and identify suspicious or ineffective members of the Oppnet.

6. *Analyzing performance of Oppnet algorithms and protocols, including the ones for localization, invitation, task offloading and coordination.* Develop measure and metrics for evaluating efficiency and effectiveness.

It would be overwhelming to study all these aspects. Therefore, our research direction and the thrust of this Thesis can be stated as:

i. Developing a proof of concept. This is a fast way to demonstrate feasibility of Oppnets.

ii. Developing a mechanism for providing services in an Oppnet by solving the Service Location-Planning (SLP) problem. This will contribute towards Challenge 4(a) (stated above), by allowing Oppnet components (seed nodes, helpers and lites) to utilize and pool resources of all nodes in an Oppnet.

Before we show in more detail how the Service Location and Planning (SLP) problem contributes towards overcoming Challenge 4(a) (from above) we will discuss the SLP problem briefly.

2.6 Service Location and Planning (SLP) Problem

In any computing paradigm, which abstracts traditional computer applications to the level of services, the challenge lies in advertising these services and optimally matching consumers, users/requesters of these services, to producers, providers/advertisers of such services. Numerous service discovery protocols, e.g. Jini, Salutation, etc. and standards, such as XML, WSDL, OWL, etc., are in existence to achieve such service advertisement and service invocation goals.

We investigate a novel service location and planning methodology that not only provides a mechanism for service mapping but also accounts for consumer-defined quality of service (QoS) parameters of throughput and delay. This is a unique methodology since to the best of our knowledge, QoS requirements have not been factored into service discovery protocols and standards.

The service location and planning methodology can be used in various computing paradigms, ranging from Cisco's latest Application Oriented Network (AON) technology [8], to the newer computing paradigm of Opportunistic Networks, to more traditional paradigms of Pervasive Computing, Service Oriented Computing, and Mobile Computing.

Let's consider an Oppnet with *m* nodes requesting a total of *t* tasks (services). In such a case, the seed in the Oppnet can poll/survey the reachable nodes in the network, indicating the QoS parameters for each of the services being requested. The QoS parameters, the cost of each service installation on every node in the network, and the underlying link layer capacities are input to the generic Service Location-Planning (SLP) Problem. The output from the SLP problem will identify the total cost of service installation to meet the demands of the network and the optimal service installation location within the network.

The SLP problem can be briefly stated as follows. Given is a network of nodes and a set of services that are being requested by nodes in the network. Associate a service installation cost with every service that is to be installed in the network to meet the requirements. These installation costs vary from one node to another. The goal is to minimize the total service installation cost, promote service federation (i.e., install multiple services on a node), meet quality of service (QoS) requirements, e.g., throughput and delay.

2.7 Oppnets and Service Location and Planning (SLP) Problem

Now, after having discussed Oppnets and the SLP problem, let us see how the SLP problem can contribute towards the challenge of optimal utilization of resources and facilities of Oppnet helpers. Let us consider the expanded Oppnet of Fig. 2 and recall that we discussed briefly how the expanded Oppnet could utilize the resources, for example, of the OnStar™ system or a Vehicular Ad-hoc Network (VANET) in the overturned car. However, the current mechanisms of service discovery and invocation cannot be applied "as is" to the Oppnet paradigm since these mechanisms require a prior or code installation on the devices. This is a feature that not only inhibits Oppnet "growth" but also inhibits most pervasive computing technologies.

With the SLP problem discussed it is possible to devise a mechanism that *provides* services in an Oppnet, so that the resources of the Oppnet can be utilized. For example, during Oppnet configuration when devices are being discovered by the seed or after configuration, the seed could probe Oppnet for the devices. Given that there is a device classification scheme, the services offered by each device can be abstracted from such a classification scheme, and then used as the input for the SLP problem. For example, in the Bluetooth medium there are methods for getting a major device class number and a minor device class number. This number identifies the device, e.g., as a computing device (using the major class device number), and as a laptop, computer, etc. using the minor class device number. Such a classification scheme could be extended so that devices in an Oppnet could not only be identified as the class they belong to but also their resources could be identified with such a scheme.

Once the device and its resources (services) have been identified by, for example, the seed, a *table of service installation costs* can be inferred. For example, let us consider the expanded Oppnet of Fig. 2, and assume that the seed used a classification scheme that enabled it to identify the devices in the Oppnet and their resources. Then, based on this information, the seed could tabulate the service installation costs as presented in Table 1. In realistic scenarios, the service installation costs would be determined based on numerous parameters such as, distant (e.g. in terms of number of hops), link bandwidth capacities, capability of device (e.g. if the device is resource—processing, storage, battery powered— constrained), line of sight (in the case of satellite communication), etc.

Table 1

Service Installation Costs for Devices in Expanded Oppnet of Fig. 2

Node in Oppnet	**Primary resources provided by node**	**Service Installation Cost for Node**		
		Communication	**Processing**	**Storage**
Refrigerator in a HAN[1]	Communication (Internet)	70	500	500
A node in a Microwave Network	Communication (Microwave Relay)	100	1000	500

[1] HAN: Home Area Network

Table 1 – Continued

Node in Oppnet	Primary resources provided by node	Service Installation Cost for Node		
		Communication	Processing	Storage
Satellite	Communication (Satellite)	110	1000	1000
A node in a Computer Network	- Computation - Communication via Internet - Storage	20	20	20
Overturned Car	Communication (via VANET[2])	50	500	500
A node in Cellular Infrastructure	Communication (Cellular)	40	100	100

The costs presented in Table 1 are however for illustrative purposes hypothetical numbers so that it can be easily inferred to which resources on which selected devices are best to be utilized by Oppnet. For example, it would be cheapest to store data/information on the computer network compared to any of the other devices, networks and systems of the Oppnets. Storing information on the cellular infrastructure will be the second cheapest system. Storing data/information on the other devices, networks and, or systems in the expanded Oppnet (cf. Fig. 2) would be more expensive than in the computer network or the cellular infrastructure since: (a) they are not intended to be used for storage, so some code installation would need to take place to enable interpreting data

[2] VANET: Vehicular Ad-hoc Network

storage and retrieval commands by the devices; and (b) in the case of non-trivial devices, like the satellite, transmission of signals would take too long and hence this delay is captured in the higher cost of service installation. Similar justifications can be made for the rest of the service installation costs that are hypothetically chosen and depicted in Table 1.

In this way, the seed is equipped with all the necessary network information such that it can map a consumer requesting a certain service, with a producer providing a certain service. For example, if there is a survivor with a bluetooth-enabled cell phone in the overturned car (in Fig. 2), she may first try calling a rescuer or emergency service, however, in the case that she does not get a cellular signal her phone would be useless to her. However, with an Oppnet operating in the area, she could search for the Oppnet and initiate a connection with Oppnet using the bluetooth capabilities of her cell phone and the bluetooth medium. She could request for communication services from the Oppnet, with this communication service, she could email her loved ones and assure them that she is alive and also convey the message that she needs help. In such a case, the Oppnet facilitates the communication by identifying the cheapest way to provide the communication, e.g. via the computer network.

In this manner, the service location and planning mechanism can improve utilization of resources in an Oppnet and thus contributes towards solving one of the challenges confronting Oppnets.

CHAPTER 3

RELATED WORK

In this chapter, first technologies related or similar to Oppnets are presented, discussed and contrasted with Oppnets, such that the novelty of Oppnets is demonstrated. Next, a review of service discovery/invocation protocols and standards, and optimization and graph theory problems that seem similar to the Service Location and Planning (SLP) problem are presented and the SLP problem is distinguished from them.

3.1 A Review of Related Technologies to Oppnets

Manish et al. [25] review technologies related to Oppnets. In this subsection we reproduce the relevant text to present a through overview of related technologies to Oppnets.

Networks can be categorized as those designed for:

- Resource-sharing, e.g. peer-to-peer and grid networks
- Monitoring and control, e.g. wireless sensor networks
- Connectivity, e.g. MANETs, mesh, ambient networks

In this section, we present a brief review of such networks.

A. Peer-to-Peer (P2P) Networks

Peer-to-peer or P2P is a subclass of distributed networks, where resource (data, bandwidth or computing power) sharing is achieved by direct exchange between the peers, rather than depending solely on centralized servers. On the basis of the level of intermediation of central servers, P2P networks can be broadly divided into pure P2P networks – those which have no central server or routers; and hybrid P2P networks – those which depend partially on central servers and routers to manage the network. In pure P2P networks the role of clients and servers is completely merged making peers with equal standing which can act as a server, client or both at the same time. In hybrid P2P networks, central servers—called supernodes or strong nodes—are used to manage information about data on peers and respond to requests for that information.

Oppnets follows similar decentralized pattern, where Oppnet-enabled devices can act as a service provider, or as a user, or as both depending on its position in a particular Oppnet hierarchy. What distinguishes Oppnets from P2P systems is the implicit heterogeneity of Oppnets due to which all the peers can not be of equal standing. Devices with limited resources and capacity (lites) such as sensing devices can perform very limited functions and can only be ordered to perform their typical task. Seeds, on the other hand, having more communicative and computing powers, may be providing entirely different range of functionalities than helpers or lites. In addition, in P2P, exchange of resources between peers is the primary goal, whereas commonality of the goal for an entire Oppnet demands much deeper coordination and collaboration among the nodes. To grow, Oppnets may follow similar methods as P2P networks, i.e. "grow by joining," however the growth of a P2P network is measured by the number of nodes joining the network, whereas the growth of an Oppnet is determined by the enhancement in the potential of the Oppnet to solve the problem at hand.

B. Grid Computing

In the beginning Grid Computing was an effort to integrate various super computers around the world but now this term has much wider implications. A computational grid basically unifies distributed computers to a single computing resource, providing users with a transparent access to the entire set of resources [9]. Users have access to the complete resource pool but not to the individual peers. IBM defines grid computing as the ability, using a set of open standards and protocols, to gain access to applications and data, processing power, storage capacity and a vast array of other computing resources over the Internet [10]. "A grid is a type of parallel and distributed system that enables the sharing, selection, and aggregation of resources distributed across 'multiple' administrative domains based on their (resources) availability, capacity, performance, cost and users' quality-of-service requirements" [11]. The term Grid as used here, is indicative of an analogy with electrical grids which provide dependable and transparent access to electricity irrespective of its origin.

Oppnets share the following characteristics with Grids [9]:

- *Multiple administrative domains and autonomy*: resources in an Oppnet, similar to Grids, may be owned by different administrative domains or different organizations. Even if they agree to be the part of an Oppnet, the autonomy of these resource owners must be honored and their local resource management and usage policies must be taken care of.

- *Scalability*: As the size of grid (and Oppnet) network grows, the problem of potential performance degradation may arise. Consequently, applications that require a large number of geographically located resources must be designed to deal with latency and bandwidth problems.
- *Dynamicity or adaptability*: Distributed networks like grid or Oppnet depend heavily on foreign resources so probability of resource failure may be high. As such applications must tailor their behavior dynamically and use the available resources and services efficiently and effectively.
- *CPU-Scavenging*: It is the process of creation of a "grid" from the unused resources in a network of participants in a grid network. To reduce burden on the volunteer helpers, Oppnets must follow similar resource utilization policies.

Although Oppnets share many features with Grids, the priorities and context of application for the two networks differ. While Grids are designed to focus primarily on computationally-intensive operations and work in homogeneous environment, Oppnets are being developed to predominantly deal with real life situations in a physical world and collaborate in heterogeneous environment. As a direct implication of this, while Grids allow remote sites to join or leave the environment whenever they choose, Oppnets may force candidate helpers to join the network in emergency situations.

C. Wireless Sensor Networks (Sensornets)

Wireless sensor networks usually consist of hundreds or thousands of tiny sensor nodes deployed across a geographical area to collectively monitor physical or environmental conditions. The nodes are small devices having one or more sensing capabilities but with very small memory and processing powers. They are also equipped with communication capabilities- usually a radio transceiver, and a power source, usually a battery. They collectively and collaboratively collect and process information, and forward it to base stations. Sensor networks are now increasingly used in environmental, military, health and home applications [12].

In typical situation, nodes are scattered in random fashion over the area to be monitored. While this allows for the deployments in inaccessible terrains, it requires that the protocols and algorithms have to be designed to provide self organizing capabilities. Due to the limitation of power sources, which may be irreplaceable, algorithms also tend to focus on high power conversion power management, in order to prolong network lifetime. Apart from these constraints, other important factors affecting the design of sensor networks are fault tolerance, scalability, operating environment, transmission media etc [12].

While lot of current research is going on in this area, little has been done to combine sensor networks with other existing networks. Since Oppnets originated from the idea of Opportunistic Sensor Networks, it is evident that Oppnets will have interfaces to sensor networks which promise to integrate their sensing and actuation capabilities to the applications when needed.

D. MANETS

Mobile ad- hoc networks consist of mobile nodes which also support routing (to cooperatively make up for the absence of fixed routing infrastructures). Transient connections are established among nodes within range and may be broken down without any prior notice or consent of all parties. Because of the dynamicity and unpredictably introduced by rapidly changing topology of the network, coupled with absence of any centralized authority, distributed operation and continuous self configuration become an essential characteristics of MANETs [13]. Due to a similar distributed nature and helper participation, continuous self configuration is also essential for Oppnets. In fact, the nature of self management in Oppnets may be more challenging as the goals of the nodes are much more than just facilitate routing. Yet another feature where Oppnets differ from traditional ad hoc networks is the level of heterogeneity. Though these networks enable heterogeneous devices or networks to communicate with each other via the common largest network, the Internet, they do not enable devices in heterogeneous communication media to communicate with each other, where as Oppnets facilitate communication across disparate communication protocols.

E. Mesh Networks

Mesh Networks [14, 18] are similar to ad hoc networks but these are supported by an infrastructure backbone provided by stationary but wireless routers. As in the case of mobile ad hoc networks, mesh networks are also self configuring and self-healing. Multiple paths ensure that mesh networks do not have to depend on any single communication link. Even if some of the connections are broken, a mesh network can still operate forming a reliable networking system. As a result, a very reliable network is formed [19].

Mesh networks can either employ full mesh topology —where each node has direct connections with all other nodes; or partial mesh topology —where every node is connected only to the nodes with which they exchange most of their data.

Typically these networks have two kinds of nodes: mesh routers and mesh clients. Although clients can also perform routing, separation of clients and routers, added by immobility of routers, make the design of protocols much simpler and cost effective as compared to mobile ad hoc networks. However, building a large-scale routing backbone is often a challenge due to scalability problems. In spite of the scalability issues, additional reliability provided by mesh networking makes it a good candidate for utilization in emergency Oppnet applications.

The contrasting feature of resource integration and utilization of Oppnets is missing in Mesh networks.

F. Ambient Networks

Ambient Networks (AN) is a European-Commission-sponsored project which aims to develop a software-driven network infrastructure that will run on top of all current or future physical network infrastructures to provide a way for devices to connect to each other, and through each other to the outside world [20]. As opposed to networking technologies mentioned above, ambient technologies do not deal with the communication link and interfaces between individual nodes but with the interfaces at the underlying network technology boundaries. The moment network boundaries are encountered; interfaces are realized by instant negotiation of agreements based on preconfigured policies.

The way Ambient Networks aim to provide end-to-end communication capabilities in heterogeneous internetworking environments may resemble how Oppnets try to integrate devices, networks or systems and manage Oppnet resources through a distributed command center. However, the following features distinguish the two efforts [20, 21, 22]:

- AN is a global, universal network intended as a replacement for the Internet (beyond 3G) and all communication networks, whereas the Oppnet is a local/wide area network meant to serve specific applications.
- AN requires heavyweight primitives whereas Oppnet requires no or only lightweight primitives.
- AN is completely pre-designed, AN is aware of the location of all sub-ANs, all its facilities are built-in or add-on, and only networks that have the needed primitives can be "composed" into ambient networks, whereas Oppnet is mostly ad hoc system that has to discover helper devices.

- ANs contact each other so that any sub-AN can initiate connections, whereas Oppnets have a mechanism where the seed Oppnet nodes initiate discovery of devices.

G. Delay Tolerant Networks

Delay tolerant networking (a.k.a. disruption tolerant networking) aims to improve connectivity between regional networks when connectivity is not continuous and prone to disruptions leading to large delays. DTN started as an effort to deal with delays in the interplanetary internet (floating nodes in space), where large distances cause much larger delays than in the earth-bound Internet, and links are disrupted for minutes or hours. The idea has been extended to other networks having similar characteristics, e.g. , terrestrial mobile networks, military battlefield networks, etc.

DTNs overcome the problems associated with large delays by adapting store-and-forward message switching [23]. DTNs also enable interoperability between different regional networks having different characteristics by providing interfaces. The storage places are capable of holding the messages for indefinite periods as opposed to holding messages for just a few milliseconds in Internet routers. In Oppnets, similar delays are expected due to two reasons. Firstly, due to high heterogeneity of the Oppnet, communication capabilities of different devices may be at diverse levels. Secondly, helpers may get disconnected due to their own constraints and workloads.

H. Class 1 Opportunistic Networks

Class 1 Opportunistic Networks (e.g., [2]) can be viewed as a generalization of the Mobile Ad hoc NETworking (MANET) paradigm.

In Class 1 Opportunistic Networks, there is no notion of utilizing resource of the nodes in the network to perform a network task. In contrast, in class 2 opportunistic networks, the network not only provides a communication backbone but can provide computing, sensing, actuating, storage, or other resources or services. Also, Oppnets can grow dynamically by admitting needed helpers, which facilitates execution of more challenging tasks. Such tasks are either beyond capabilities of traditional networks, or are much more difficult to achieve even in Class 1 Opportunistic Networks. As shown in Appendix A, Oppnets provide higher-level primitives to facilitate building of complex applications.

Class 1 Opportunistic Networks are a proper subset of Class 2 Opportunistic Networks or Oppnets.

I. Spontaneous Networks

Spontaneous networking is a relatively new area of research focusing on a small subspace of ad hoc networks. Aim of the network is not just providing connectivity but supporting the collaborative activity of a group of devices supported by wireless communication [24]. They resemble Oppnet in some of their features, including heterogeneity of nodes and collaboration among the participants. However, hierarchy of nodes in Oppnets and administrative capabilities of seed nodes is not found in spontaneous networks. Also, while spontaneous networks are based on the physical proximity of a restricted number of nodes located nearby each other, Oppnets may grow considerably according to the needs of their tasks.

3.2 Qualitative Comparison of Related Technologies with Oppnets

In this section, a qualitative comparison of the related technologies discussed in the previous section is presented. The tables originally appeared in Lilien [17] and Manish et al. [25].

Table 2

Comparison of Features of Selected Networks – Part I [25]

Feature	subfeatures (if any)	P2P Systems	Computational Grids	Sensornets	MANETS	Mesh Networks
Distinguishing Features (w.r.t. other ad hoc nets)		domination of peer nodes (over clients or servers), resource aggregation	lightweight nodes with sensors, energy-constrained, densely deployed	Virtual pool of resources, users have access or knowledge of the pool and not of nodes	rapidly changing topologies, lack of centralized entity	some hosts are also routers, lack of centralized entity, very reliable
Deployment	Rapid	Yes	Yes	yes	yes	yes
	incremental	in principle, yes	Yes	possible	yes	yes
Configuration	limits on minimal required configuration	starts with a seed	Co-ordination among nodes required	no	no	No
	Nodes join/leave	often (even w/o warning)		no	often (even w/o warning)	infrequently (once established)

Table 2 – Continued

Feature	subfeatures (if any)	P2P Systems	Computational Grids	Sensornets	MANETS	Mesh Networks
Operation	self-organizing	Yes	Yes	Yes	yes	yes (routers)
	standalone/ connected to Internet	Yes/Possible	Yes/Possible	no (requires remote task manager)/ yes	Yes/Possible	Yes/Possible
	centralized entities	some (e.g., DNS)	in the form of Administrative hierarchy	sinks or base stations, gateways	None	None
	resource aggregation	Yes	Yes	No	No	No
Node Types		"pure" peer nodes, supernodes or super-peers and client peers	base station or sink, sensor nodes (possibly with routing capabilities)	Computational devices creating pool of computational resource	mobile, stationary (few)	mesh routers, mesh clients, conventional clients
Node Characteristics	lightweight nodes	No	No	mostly	possible	as clients
	software heterogeneity	low	No	No	low	Low
	hardware heterogeneity	high	Possible	No	high	High
	Limited energy	No	No	Yes	some	Possible
Node Examples		desktop, laptop	mote, Internet gateway node	super, cluster and ordinary computers/laptops, PDA	pedestrians, soldiers, unmanned robots, vehicles, buildings	desktop, laptop, PDA, WiFi IP phone, RFID reader

Table 2 – Continued

Feature	subfeatures (if any)	P2P Systems	Computational Grids	Sensornets	MANETS	Mesh Networks
Node Mobility	Stationary nodes	yes	possible	Yes	possible	Possible
	Mobility of combined hosts/routers	D.N.A.	No	No	high	D.N.A.
	Mobility of separate hosts (clients/peers)	possible	possible	D.N.A.	D.N.A.	Yes
	Mobility of separate routers	D.N.A.	possible	D.N.A.	D.N.A.	Minimal
Communication	wireless/wired	yes/yes	yes/yes	yes/gateways to wired	yes/possible	yes/possible
	Limited bandwidth for some nodes	yes	No	yes	yes	Yes
	persists once established	no	Yes	yes	no	Yes
	connection to Internet	typical	Mostly	possible, via base station	possible	possible
	limited node transmission radius	yes if wireless	yes	yes	yes	yes (wireless mesh routers)
	most communicat-ion between nearby nodes	yes	no	yes	yes	Yes
	routing by	by underlying network (e.g., Internet)	By underlying network	usually, by base stations or cluster heads; by sensor nodes also possible	all nodes	by mesh routers
	interoperability	no	possible	usually via gateways	possible	Possible

Table 2 – Continued

Feature	subfeatures (if any)	P2P Systems	Computational Grids	Sensornets	MANETS	Mesh Networks
Topology	arbitrary	yes	Yes	yes	yes	yes
	time varying	highly	Yes	yes (mostly due to using up energy, jamming or noise, moving obstacles)	highly	limited
	typical size [in nodes]	even millions	K*10-k*1000	even millions	k*100-k*1,000	arbitrary
	typical area covered	MAN, countrywide	LAN-like or MAN-like	LAN-like or MAN-like	MAN, countrywide	LAN, MAN

Table 3

Comparison of Features of Selected Networks – Part II [25]

Feature	subfeatures (if any)	Ambient Networks	DTN	Class 1 Opportunistic Networks	Spontaneous Networks	Oppnets
Distinguishing Features (w.r.t. other ad hoc nets)		no administrative infrastructure, human interaction and proximity leveraged for configuration	Interoperation of heterogeneous networks major goal, deals with technology boundaries	no administrative infrastructure, human interaction and proximity leveraged for configuration	tactical resource aggregation, high software heterogeneity, high interoperability	no administrative infrastructure, human interaction and proximity leveraged for configuration
Deployment	Rapid	D.N.A.	Yes	yes	yes	yes
	incremental	D.N.A	yes	yes (from the seed)	yes	yes (from the seed)

Table 3- Continued

Feature	subfeatures (if any)	Ambient Networks	DTN	Class 1 Opportunistic Networks	Spontaneous Networks	Oppnets
Configuration	limits on minimal required configuration		Should support packet switching	starts with a seed	no	starts with a seed
	nodes join/leave	D.N.A	Yes	mostly during growth or dismantling	infrequently (except at session/ subsession beginning/ end)	mostly during growth or dismantling
Operation	Self-organizing	D.N.A	---	yes	yes	yes
	standalone/ connected to Internet	no/ Yes	yes/ No	yes/ possible	yes/ no	yes/ possible
	centralized entities	some (e.g., DNS)	None	none or in the seed	none	none or in the seed
	resource aggregation	no	no	no	no	yes
Node Types		peer nodes	"pure" peer nodes, supernodes or super-peers and client peers	peer nodes	seed nodes, helpers	peer nodes
Node Characteristic	lightweight nodes	possible	No	possible	no	possible
	software heterogeneity	high	No	high	no	high
	hardware heterogeneity	high	possible	high	possible	high
	limited energy	No	possible	infrequent	no	infrequent
Node Examples		laptop, PDA, high-end mobile phone	Does not deal with nodes	laptop, PDA, high-end mobile phone	desktop, laptop, PDA, WiFi IP phone, mote, RFID reader	laptop, PDA, high-end mobile phone

Table 3 – Continued

Feature	subfeatures (if any)	Ambient Networks	DTN	Class 1 Opportunistic Networks	Spontaneous Networks	Oppnets
Node Mobility	stationary nodes	Possible	no	yes	yes (when in use)	yes
	mobility of combined hosts/routers	Possible	No	yes	no (when in use)	yes
	mobility of separate hosts (clients or peers)	Possible	yes	yes	D.N.A.	yes
	mobility of separate routers	possible	yes	yes	D.N.A.	yes
Communication	wireless/wired	Yes/ yes	Yes/ No	yes/ yes	yes (short-range for session establishment, then regular) /unlikely	Yes/ yes
	limited bandwidth for some nodes	D.N.A	yes (connection in bits)	yes	yes (session setup)/no (session)	yes
	persists once established	D.N.A	no	yes	yes	yes
	connection to Internet	typical	no	possible	unlikely	possible
	limited node transmission radius	Yes if wireless	yes	yes if wireless	yes	yes if wireless
	most communication between nearby nodes	no	Not necessary	maybe	yes	maybe
	routing by	By underlying network	All nodes	by most nodes or by routers	all nodes	by most nodes or by routers
	interoperability	yes	No	high	no	high
Topology	arbitrary	Yes	yes	yes	yes	yes
	time varying	yes	possibly	mostly during growth or dismantling	mostly at the beginning or end of session or subsession	mostly during growth or dismantling
	typical size [in nodes]	even millions	few	arbitrary	small to medium	arbitrary
	typical area covered	planetary	interplanetary	MAN	one room	MAN

3.3 Literature Review for Service Location-Planning (SLP) Problem

In this section we discuss various optimization problems that are related to or similar to the SLP problem.

The Capacitated Facility Location (CFL) problem can be stated as follows [56], given a set of facilities F and a set of clients C, each $c \in C$ has a demand dc that must be serviced by one or more open facilities. There is a facility cost fi for opening facility $i \in F$ and a service cost si,c for facility i to service one unit of demand from client c. No facility may service more than U units of demand. The goal is to service all clients at minimum total cost (i.e. sum of facility and service costs). The CFL problem is NP-Hard even in the case where $U = \infty$, known as the Uncapacitated Facility Location (UCFL) problem [26, 27, 56].

In the SLP problem, there is another dimension to the problem, the QoS constraint to meet delay. This makes the SLP problem more complex than the Facility Location (FL) problems.

Other problems of interest, such as optimal placement of gateways in wireless mesh networks [28], [29] and service selection to minimize cost or maximize QoS [30], are also considered variants of the Facility Location problem.

Aoun, et al. [28] consider the problem, that given a wireless mesh network, how do we place gateways optimally that connect to Access Points (AP), such that the number of gateways are minimal, subject to delay, throughput, bandwidth. The authors factor, delay as number of hops, throughput is simplified to congestion and bandwidth to cluster size. They use a cluster approach so that nodes are encompassed in disjoint clusters.

However, the mesh gateway placement problem does not take into account communication delay or other constraints such as the underlying link layer constraints that are contained in the SLP problem.

Bejerano [29] considers the same mesh gateway placement problem [28], however the solution approach is different, as the author solves gateway placement cost and clustering separately.

The authors in [30] approach a similar problem, rooted at FL problem, from a service selection point of view, such that requests are "binded" to services that minimize cost, or maximize QoS constraints. Again, the freedom of service placement on location makes our problem different. Also, authors in [59] study various approximation algorithms for the facility location problems, that seem to be the root of related work presented here, that most closely resembles our problem.

From Graph Theory related optimization problems, it seems that the Vertex Cover problem and its variants are similar to the SLP problem. However, a close look at the Vertex Cover problems shows that firstly, it does not allow formulation of QoS constraints. Secondly, Vertex Cover problems in undirected graphs [31] cannot help in the formulation of the SLP problem since they are in undirected networks.

However, Vertex Cover problems in directed graphs [32] can be considered in future work as the basis for heuristic and meta-heuristic programming for SLP problem.

Furthermore, these problems cannot be used to guarantee optimal solutions for the SLP problem, where as the Integer Linear Programming (ILP) model of the SLP problem (that we formulate and discuss in this thesis) gives optimal solutions for the SLP problem (shown in later chapters).

A review of Networking related optimization problems shows that those problems pertaining to service or content replication are similar to the SLP problem. For example, Jin and Wang [57] reduce communication overhead by replicating content and services in differently from the traditional pull mechanism practiced. In the pull mechanism, replicates of an object are proportional to the popularity of the object. This approach can be used in future work to analyze the tradeoff between service replication and service installation.

Another alternate to service replication is service composition as studied in [58]. However, in SLP we account for service composition through service federation constraints.

Other technologies, e.g. Service Discovery Protocols (SDP) from Pervasive Computing and Service Oriented Computing paradigms, can be studied to gain insight for resource utilization and invocation. Zhu and et el. [33] present a comparison of existing protocols that implement service discovery and communication—JiniTM, UPnPTM and Bluetooth Service Discovery ProtocolTM (SDP)—to mention a few. A striking difference, however, between all these service discovery protocols and those needed in the more recent networking technologies such as those of Oppnets, AONs, etc. is the need for pre-configuration of the devices with the common protocols. Oppnet is facilitation by "training" but able to function with any pre-configuration other than the seed Oppnet pre-configuration. This restriction also inhibits creating truly pervasive and ubiquitous computing environments.

What is lacking in these optimization problems, graph-theory problems and service discovery protocols necessitates consequentially the novel Service Location-Planning problem. The new problem not only meets consumer demands for services with QoS parameters but minimizes service installation cost for the producers. All this while operating within the limits of the underlying network link layer bandwidth capacities.

The Service Location-Planning (SLP) technique can be used in the latest technology in Service Oriented Computing of Application Oriented Networks (AON), launched by Cisco Systems, a leading communications technology and infrastructure provider. The AON Technology enables relocation of application services from end nodes (end-points) in the network to the routers and switches in a network [8]. This is a huge impact technology since it allows application-level message intelligence at the router level.

This has two obvious fundamental consequences. First, AON routers can route application-layer messages, such as stock quotes, weather and news alerts, etc., to the appropriate application service rather than an arbitrary IP address. This is done through "bladelets", a set of operations, which are applied to application messages. Second, services can be installed on AON routers. This enables faster satisfaction of service demand requests, as these requests are not propagated to the end-points (outside) of the network, rather interpreted and satisfied within the network.

CHAPTER 4

MICROOPPNET – A SMALL-SCALE OPPNET

We have published the design and implementation of the small-scale Oppnet, called MicroOppnet [25]. I reproduce the relevant text in this chapter to discuss the design and implementation of MicroOppnet and a sample scenario that benefits from MicroOppnet.

4.1 Overview of MicroOppnet

The current version of the MicroOppnet, is a small-scale proof of concept and test bed for Class 2 Opportunistic Networks, since it not only allows opportunistic communications but also opportunistically accesses sensornet (sensor network) nodes to perform sensing. Since sensing is the only "class 2" activity, it is rudimentary in its class 2 opportunism. Hence the prefix "micro" in the name "MicroOppnet."

The MicroOppnet is a platform on which *functional components*, such as, Oppnet components, primitives, protocols, and architectures are or will be implemented, tested, and fine-tuned. *Non-functional parameters*, including quality of service (QoS) parameters, such as throughput, delay, reliability, accuracy, scalability, etc., can also be investigated on the MicroOppnet.

Figure 4, shows the structure of the seed Oppnet in the MicroOppnet consists of Workstation A with a Bluetooth (BT) adapter and a serial port connection to Sensornet

Base Station BS_1. The seed searches for BT devices and initiates a connection with them. Alternatively, a BT-enabled device—a cell phone labeled Victim in our example—can find the seed and initiate a connection once a connection has been established, the Victim cell phone can send a message to the seed, for example, the help message.

This message is then forwarded via Base Station BS_1, and then through the sensor network. In the MicroOppnet, the sensornet consists of 10 Mica2 Motes and 6 Stargates, which are sensornet gateways. Some of the gateways are also connected to Mica2 Motes.

Base Station BS_2 at the other end of the sensornet is connected to Laptop B. Once the help message is propagated via BS_2 to Laptop B, a Java TCP/IP client socket connection is initiated with a remote Java server. The help message and the location of the device that sent it are logged on this server.

The Java server can be queried by remote users employing either traditional computing devices or Java-enabled devices. In our example, we employ cell phone with T-Mobile™ Virtual Private Network (VPN) connection, labeled as Responder.

The seed can broadcast to the sensornet a variety of messages in addition to help—e.g., start_sensing, log_sensing, retrieve_log. The messages can be used, for instance, to start temperature sensing, to log temperature in the EEPROM of the sensor, or to retrieve the logged data from the sensor network. The retrieved temperature readings can be logged at the Java server. Then, they can either be queried by remote users via wireless Internet, or be broadcast by the seed on the BT channels.

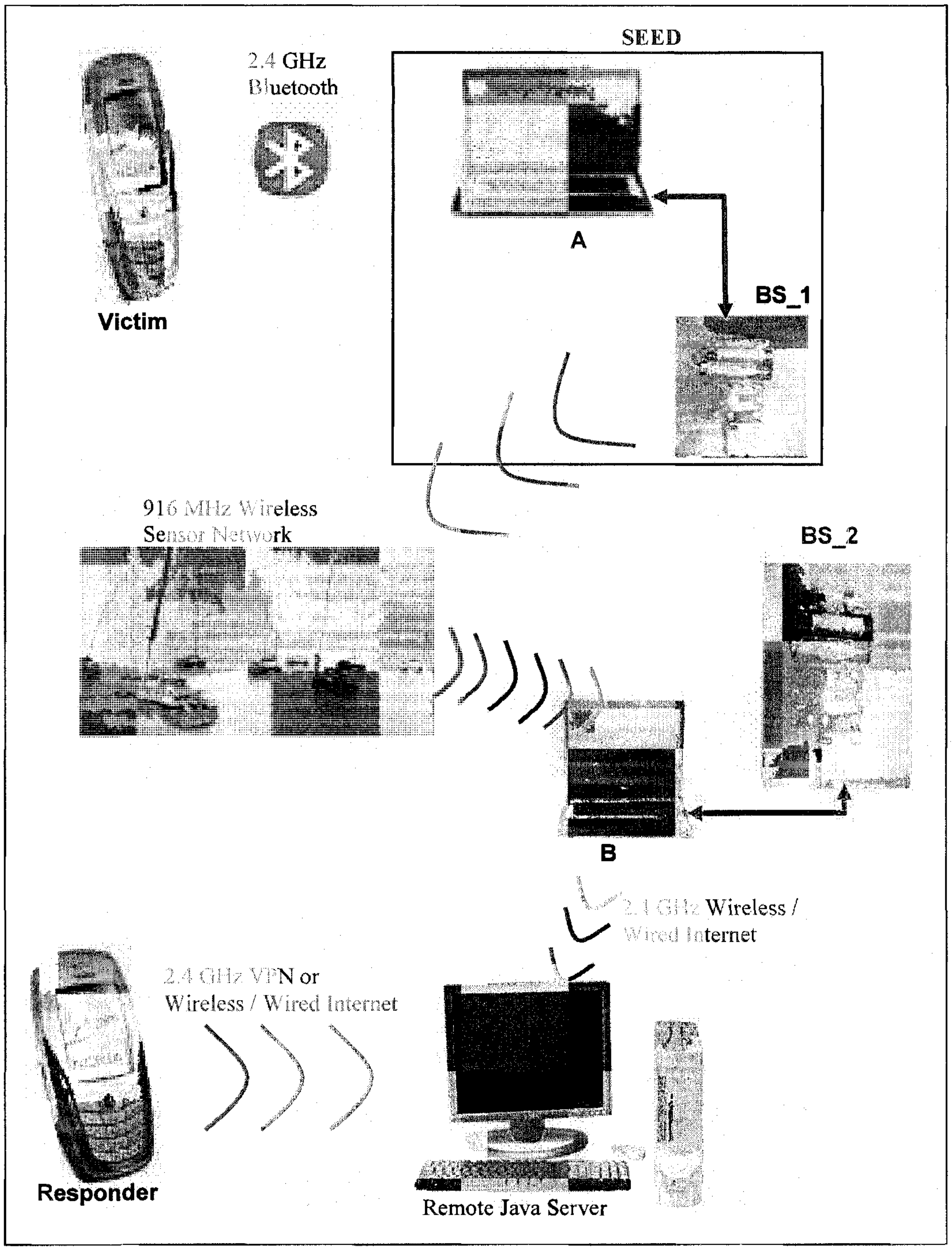


Figure 4. Structure of MicroOppnet v.2.2

The current version of the MicroOppnet, integrates only three disparate communication media and frequency ranges, namely, BT (2.4 GHz), a sensor network (916/896/433MHz), and the wireless Internet 802.11b and 802.11g (2.4 GHz [13], the same frequency as BT).

4.2 Design Choices

There were numerous design choices in the design and implementation of MicroOppnet. For example, we have considered a number of technologies, such as Wired/Wireless Internet, InfraRed, Microwave, WiMAX, Satellite, Cellular networks, unrestricted and standardized Bluetooth medium and the radio frequency of 916 MHz of the Wireless Sensor Network, etc., that could have been used in the implementation of the MicroOppnet. Our decision criteria were ease of operation and availability.

With these design criteria in mind, the obvious choice would include the traditional wired Internet and its wireless counterpart. The readily available hardware (e.g. USB Bluetooth adapters/converts, Bluetooth enabled devices, etc.), protocols and standards (e.g. Java Bluetooth API, JSR-82, Atinav AveLink Bluetooth Protocol Stack [34]), and good technical support and documentation for Bluetooth medium made it an easy and economical choice for experimentation. Similarly, the availability of sensornet devices and my experience with sensornet programming made this technology a component for MicroOppnet experimentation.

A summary of the project goals, evaluation criteria and decisions made are presented in Table 4 [53].

Table 4

Requirements of MicroOppnet, Design Criteria and Decisions [53]

	Project Goals	Evaluation Criteria	Design Choices	Decision
1	Integrate disjoint communication media, protocol and, or standards	• Availability • Ease of Use • Dependability • Acceptability • Popularity	• Wireless Internet • Wired Internet • Infrared • Bluetooth • WiMAX • Sensornet • Satellite • OnStar™ • Cellular Network • Telephony/VoIP	• Wireless Internet • Wired Internet • Bluetooth • Sensornet • Cellular Network
2	Minimal a priori code installation	• Flexibility • Reusability	• SDP[3] Protocols – Jini, Salutation, etc. • Naming Schemes • Data Formats	• Bluetooth SDP • Existing naming and addressing schemes and data formats
3	Class 2 opportunism–leverage resources	• Ease of use • Availability	• Computation • Sensory • Actuation • Storage	• Sensory actions and records, since sensornet is used

4.3 Design of MicroOppnet

In this section, we present the flow of control for the MicroOppnet of Figure 4 in terms of the Oppnet Virtual Machine (OVM) primitives. The OVM is a standard implementation framework for Oppnet applications, so that there is interoperability of

[3] SDP – Service Discovery Protocols

application programs, hardware devices, environments and tools [6]. (See Appendix A, for a detailed discussion of the OVM and its primitives)

The flow of control, illustrated in Figure 5, can begin with: (a) an active discovering of candidates—using the OVM primitive SEED_discover; (b) with a passive wait—using SEED_listen, when candidates search for and initiate connection with the seed; or (c) with dispatching a task for the sensornet—using SEED_sendTask. In the MicroOppnet, communication for (a) and (b) is only over the Bluetooth medium.

Messages received from nodes wishing to use the MicroOppnet are processed, and tasks are delegated to the appropriate helpers. In the MicroOppnet, there are only two sets of helpers: the set of nodes in the sensornet, and the remote server.

Messages from a user such as Victim in Figure 4 can be forwarded from the seed's sensornet Base Station (SBS) BS_1 to the helpers using the SEED_sendTask primitive. The nodes in the sensornet process the message using HLPR_processMsg and then perform the task (currently, only sensing or communication) using HLPR_runApplication. If the task is sensing, then the sensornet nodes (SNNs) will start or stop sensing as required. Otherwise, they will forward either the received message or their temperature sensor readings as directed. When the message is received by another sensornet gateway or another base station (e.g., by BS_2), it is logged on a remote server. If the task was to retrieve sensor-measured temperature, then BS_2 aggregates sensornet readings and floods the result back through the sensornet to BS_1.

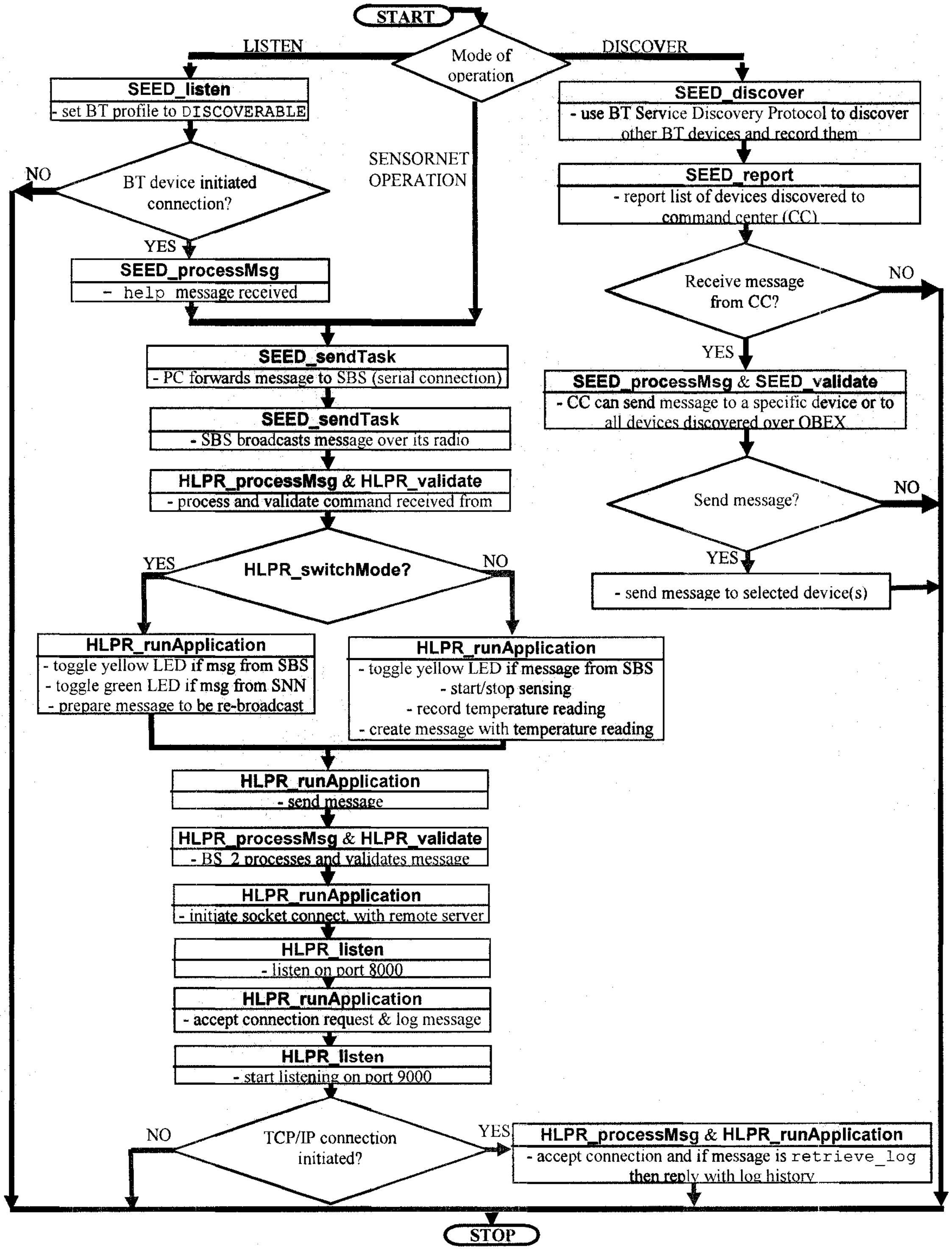


Figure 5. Flow of control in the MicroOppnet v.2.2

Devices such as Responder (cf. Fig. 4) can send the message retrieve_log to the remote helper server, which is in a listening mode with the HLPR_listen primitive. This allows the remote server's log to be queried for specific tasks and retrieve the appropriate messages. The server can process any TCP/IP socket connection.

Summarizing, the MicroOppnet supports only two categories of tasks: (i) communication tasks – flooding messages and retrieving sensor readings; and (ii) sensing tasks – starting and stopping sensing.

All these tasks rely on opportunism. In more detail, the following is the exhaustive list of all tasks using resources opportunistically:

- Communication in the BT medium
- Communication in the sensornet medium
- Communication using TCP/IP in wired or wireless Internet
- Temperature sensing using sensornet nodes

The first three tasks use Class 1 opportunism, and only the last task relies on Class 2 opportunism—by leveraging the sensing resources of MicroOppnet helpers. Thanks to the last task, we can claim that MicroOppnet is a Class 2 Opportunistic Network, albeit a rudimentary one (exploiting only one type of non-communication resources).

4.4 Implementation of MicroOppnet

A USB Bluetooth (BT) dongle equips the seed with a BT infrastructure. To exploit the BT communication framework, we use the BT software protocol stack provided by Atinav AveLink [34]. In this way, we can invoke the BT Service Discovery Protocol (SDP) using the API of the protocol stack to detect BT devices, and to either initiate connections with BT devices or to receive connections from BT devices.

The BT communication infrastructure consists of profiles that are built on top of layers/protocols to define further high-level functionality. There are numerous profiles that exist and, moreover, there are close dependencies between profiles. The lowest-level profile that most common BT Profiles are dependent on is the Generic Access Profile, which is used to establish a basic connection. After establishing an initial connection, we use Generic Object Exchange Profile, which uses the Object Exchange (OBEX) layer to exchange objects. Alternatively, we can use Logical Link Control and Adaptation Protocol (L2CAP) and RFCOMM protocol (uses Serial Port Profile) for packet and stream data, respectively [35].

Our sensor network consists of Crossbow's Mica2 Motes and Stargate gateways [36]. The Mica2 Motes run UC Berkley's TinyOS [37] operating system, and are programmed with nesC [38]. The nesC code is compiled on a workstation and is flushed onto the Motes using Crossbow's programming boards.

The remote server, developed in Java using socket connections, runs on a Linux machine. Its flow of control is illustrated in Figure 6.

Cell phone programming is accomplished with Java MicroEdition (J2ME) and JSR-118 Mobile Information Device Profile (MIDP) 2.0 for resource-constrained devices, such as cell phones and PDAs. Java applications for such devices are called MIDlets.

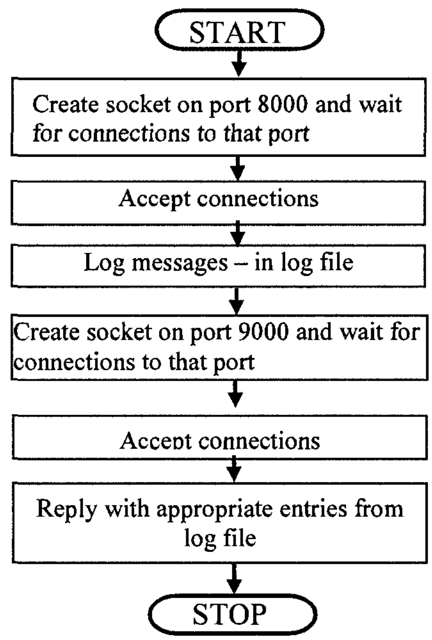


Figure 6. Flow of control for the remote Java server

Figure 7 illustrates the flow of control in the MIDlet of the Responder (cf. Fig. 4). It should be noted that the Victim (cf. Fig. 4) does not need any priori code installation. Java-enabled phones, specifically Nokia 6600 (equipped with Symbian OS), Nokia 6103, and Motorola RAZR were used in this implementation. And it was found that Nokia 6600 is stronger than the other two models when it came to initiating BT connections with the seed or TCP/IP connections with the server.

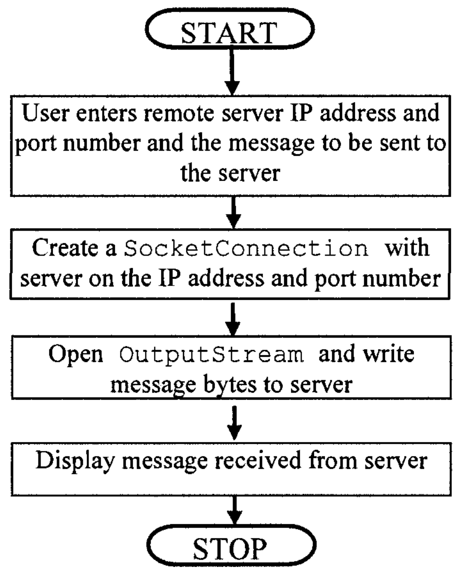


Figure 7. Flow of control for cell phone MIDlets

In this version of the MicroOppnet, a MANET routing approach is used, in which every node in the Oppnet is a router.

The footprint (in Kilobytes) of the individual applications that are installed on the various devices depicted in the MicroOppnet are as follows [53]:

1. Code on the Victim cell phone: There is no code installation on the Victim cell phone, since any cell phone with a Bluetooth connection can use the Bluetooth Service Discovery protocol (already installed in all Bluetooth-enabled phones) to search for other Bluetooth devices. This service discovery is able to show Oppnet Seed as one of the potential devices, and then ask the user of the Victim cell phone whether connection should be initiated with the seed.
2. Oppnet Seed code:

   a. Laptop A: 17 KB, and
   b. Sensornet base station BS_1: 18.4 KB
3. Oppnet helper code:
   a. Wireless Sensor Network nodes: 19.6 KB
   b. BS_2: 18.4 KB, and
   c. Laptop B: 2 KB
4. Code for Remote Java Server: 2 KB
5. Code on the Responder cell phone: 4 KB

The MicroOppnet code is given in Appendix B.

4.5 Challenges in Developing MicroOppnet

In this section we delineate the hardware and software challenges we met in implementing the MicroOppnet.

4.5.1 Hardware Challenges

Nokia 6600 was a much more robust cell phone, probably due to its Symbian OS, since MIDlets that created sockets or streams were not allowed on Nokia 6103, locked by T-Mobile and Motorola RAZR was not equipped to receive text messages over Bluetooth, invalid format errors.

4.5.2 Software Challenges

1. T-Mobile WAP (Wireless Application Protocol) and GPRS (General Packet Radio Service) connections do not allow unrestricted access to the Internet,

instead T-Mobile Virtual Area Network (VPN) was used to allow unrestricted MIDlet access to the Internet.

2. The remote server should not be behind any firewall to allow MIDlet access to the server.

4.6 Sample Application Scenarios for MicroOppnet

To illustrate use of the MicroOppnet, let us consider an emergency scenario, namely a fire in a large office building. Suppose that some workers were unable to evacuate. Most of them tried to use their cell phones to call for help. Many succeeded but many failed to get a connection since the cell phone infrastructure is overloaded with calls being made by thousands of workers still gathered outside of the building.

The firefighters can put a MicroOppnet (or, maybe, a MiniOppnet) to use. They deploy around the office building the MicroOppnet seed, consisting of laptops and networks connecting them. Now, the Bluetooth (BT) Class 1 connectivity (BT Class 1 has the range of approx. 100 meters) becomes an essential communications capability, with the MicroOppnet using it to discover all kinds of BT-enabled helpers. An owner of any such helper, that is an owner of a BT-equipped cell phone, PDA, laptop, etc., is now able to communicate with the firefighters via the extended MicroOppnet (consisting of the seed MicroOppnet plus all helpers that joined it).

This only illustrated Class 1 opportunistic capabilities of the MicroOppnet. To show how Class 2 opportunistic capabilities of the MicroOppnet can be used, suppose that the MicroOppnet is now commanded to contact and query for temperature readings from all sensing nodes within the building (they include a multitude of Oppnet-enabled, in this case BT-enabled, smoke detectors with add-on multisensor capabilities). These temperature readings, aggregated at a Java server, are used to plot the heat profile for the building. The profile, together with location information gathered by BT-equipped helpers before, can be used by the firefighters to find the best routes for reaching the workers trapped in the building by fire.

Note that many other pervasive communications technologies could be used in parallel with BT.

CHAPTER 5

THE SERVICE LOCATION-PLANNING (SLP) PROBLEM

A page on the Service Location and Planning (SLP) problem, for small-scale networks presented in this chapter has been published in [40]. We reproduce the relevant text here to discuss the SLP problem.

5.1 Service Location and Planning Problem – A Formal Definition

The Service Location and Planning (SLP) Problem can be defined as follows.

Given: A network graph G=(V, E) and a set of services S, where V is set of vertices/nodes and E is the set of edges. There exists a set of consumers, $V^C \subseteq V$, such that all nodes $v \in V^C$ are requesting a service(s), $s_i \in S, i = 1..|S|$ and have a throughput and delay demand associated with each request. There is also a service installation cost associated with each service s on a node n in the network G and a service discount Γ is given to promote service federation, that is, multiple service installations on the same node.

The cost of installing a service s on a node n can include: (i) cost of accessing the node, i.e. either physically/manually or in terms of number of hops, (ii) storage capacity of the node, (iii) processing capacity of the node, (iv) installing the application (i.e. the service s) on the node n, etc.

Installing multiple services on a node is promoted because based on the service installation costs, it will be beneficial to a producer *p*, if its service *s1* is installed on node *n* at the same time when another producer *q*'s, service *s2* is being installed on *n*. This way, at the least the cost of accessing *n* is shared between the two producers, which can be formulated as a discount in the service installation cost.

Problem: Install services on a set of producers $V^P \subseteq V$, such that the service installation cost incurred is minimal and all throughput and delay requirements are satisfied, while also satisfying the underlying link layer capacities. Note: producers can also be consumers of services $\Rightarrow V^P \cap V^C \neq \emptyset$.

5.2 Methodology Overview

Mathematical formulation of research problems as optimization problems is a research methodology that is used in diverse areas of research and study ranging from economics to physics to computer science. When a research problem can be successfully formulated as a mathematical problem/model, it can be expressed and defined unambiguously and solved accurately.

However, often hard or complex formulations cannot be solved effectively or are computationally intensive. Furthermore, some mathematical formulations can only be solved on a small-scale and when large-scale scenarios are considered, the problem becomes too large or complex to be solved using traditional ILP/LP solve engines (e.g.

CPLEX, LpSolve [39], etc.). In such cases, approximation techniques are used to solve the problem.

The methodology adapted for this phase of the research is as discussed above. First, a formal definition of the problem, followed by the definition of a mathematical model to represent the problem precisely, accurately and unambiguously, in the form on an Integer Linear Programming (ILP) Problem. Next, solving this ILP using traditional ILP/LP solve engines to achieve optimal results for small-scale networks and then application of Lagrangean Relaxation technique for the approximation of the SLP problem for large-scale networks.

5.3 Assumptions in the Formulation of the SLP Problem

Delay in a network is attributed to queuing delay at intermediate nodes, propagation delay and transmission delay. These can be formulated as $T_{queuing} = \frac{1}{\text{link capacity} - \text{load on link}}$, $T_{transmission} = \frac{\text{data size}}{\text{link capacity}}$ and $T_{propagation} = \frac{\text{link length}}{\frac{2}{3}c}$, where c is speed of light and the queuing delay is for M/M/1 queues [54]. Consequentially, end-to-end delay formulation is non-linear, due to the non-linearity in end-to-end queuing delay. Note accounting for transmission and propagation delay is trivial and not part of our formulation.

We use an approximation technique to formulate linear queuing delay constraints. The technique is simple. We use a load-delay lookup table to compute the non-linear queuing delay at a link, by calculating the load on the link, and looking up the delay value for that load on that link in a load-delay lookup table. In Fig. 8 we illustrate a load-delay lookup tables for bandwidth capacities of 100, 300, 600 and 1000, this implies that as long as the maximum bandwidth capacity in the network being considered is 100, 300, 600 or 1000 Mb/s we can approximate the queuing delay at the respective link by looking up the load on that link.

The entries in the load-delay lookup table are computed by adapting the queuing delay equation presented above $\left(T_{queuing} = \frac{1}{\text{link capacity} - \text{load on link}}\right)$. For example, when link bandwidth capacity is 100 Mb/s and there is a load of 50 Mb on this link, then the queuing delay is, $T_{queuing} = \frac{1}{\text{link capacity} - \text{load on link}} = \frac{1}{100-50} = \frac{1}{50} = 0.02\,\text{sec}$. We modify this queuing delay equation so as to avoid working with decimal numbers, by computing queuing delay according to $T_{queuing} = \frac{1}{\text{link capacity} - \text{load on link}} \times 10{,}000$. In this case, the queuing delay on a link with capacity of 100 Mb/s and 50 Mb, is $T_{queuing} = \frac{1}{\text{link capacity} - \text{load on link}} \times 10000 = \frac{1}{100-50} \times 10000 = \frac{1}{50} \times 10000 = 200\,\text{sec}$. Note, that when the load on a link is maximum, e.g. link capacity 100 Mb/s and load 100 Mb, then queuing delay, according to our modified $T_{queuing}$ equation is $T_{queuing} = \frac{1}{100-100} \times 10000 = \frac{1}{0} \times 10000 = nan$, an undefined number. However, logically any link can support maximum capacity, though the delay on such a link would be large, largest of queuing delays of all loads less than maximum capacity. Thus, we arbitrarily specify this maximum queuing delay to be 500 sec. on any link with maximum capacity load.

So, now for a link with bandwidth capacity 1000 Mb/s and load 50, the queuing delay can be approximated as 9 sec. In this case of the load-delay lookup table of Fig. 8, the load lookup interval is 50. This implies, that if, a link with capacity 100 Mb/s had a load of 30, the delay would be undefined, since the table only delineates delay for loads of 0, 50 and 100 (for 100 Mb/s link). We round up the load on a link to the nearest lookup value, so that such cases are defined. For example, for a link with bandwidth capacity of 600 Mb/s and load of 137, in the ILP formulation we round up the load so that the delay on this link is equivalent to the delay with load of 150, that is 22 s.

Therefore, there is a tradeoff between bandwidth wastage and lookup table storage. If the lookup table has unit increments of load then the lookup table will be huge to allow lookup for all loads ranging from 0,1,2...*max_bandwidth*, since *max_bandwidth* also caps maximum load on a link in the network. On the other hand, if lookup table is in increments of, say 100, then we are wasting bandwidth since delay for load of 225 will be undefined and load will have to be rounded up to 300 for a load-delay lookup in the table.

However, to overcome the nonlinearity in delay formulations we compromise with bandwidth wastage. Some of the load-delay lookup tables used in our implementations are illustrated in Fig. 8, with load interval of 50 in the delay lookup table.

Consequentially, we realize that bandwidth resources are wasted, if a link with load=225 is rounded up and delay for load 300 is assigned to this link. However, this is a tradeoff we make to formulate linear delay constraints. Furthermore, the bandwidth wasted is directly proportional to the scale used in the lookup table, which in turn is directly proportional to the storage required for the lookup table.

A second assumption we make to allow local service installation is that all nodes have a zero-cycle loop with *max_bandwidth* of 1000 Mb/s. In this case, when needed services can be stored locally.

Link bandwidth capacity = 100

Load	Delay
0	0
50	200
100	500

(a)

Link bandwidth capacity = 300

Load	Delay
0	0
50	40
100	50
150	67
200	100
250	200
300	500

(b)

Link bandwidth capacity = 600

Load	Delay
0	0
50	18
100	20
150	22
200	25

Link bandwidth capacity = 1000

Load	Delay
0	0
50	9
100	10
150	11
200	12

250	29
300	33
350	40
400	50
450	67
500	100
550	200
600	500

(c)

250	13
300	14
...	
800	50
850	67
900	100
950	200
1000	500

(d)

Figure 8. Load-Delay Lookup Tables for bandwidths of 100, 300, 600 and 1000 Mb/sec

We promote service federation, so that if the cost of installing service 1 of a node X is 100 and the cost of installing service 1 on a node Y is 80. Also assume a service 2 is already installed on node X for a cost of 60. Assume there is a "discount" of 20 given for multiple services installed on a node.

In this case, if we install service 1 on node Y and service 2 on node X, then the total cost of service installation: 60+20 (on node X) + 80+20 (on node Y) = 180. However, if service 1 and service 2 are installed on node X, then the total cost of service installation: 60+80+20 (on node X) = 160. This implies, though the cost of service installation on node Y for service 2 is less than that on node X, the total cost of service installation is less when service 1 and 2 are installed on node X.

The motivation behind service federation is that if a node X must be manipulated to offer a certain service then it is more feasible to enhance it simultaneously for another service rather than to manipulate another node to equip it for the same service. Simply stated, consider calling a plumber to fix a leaking kitchen sink and then calling another plumber to fix the bathroom, rather, it is more logical, feasible, and economical to call one plumber that can fix both the kitchen and bathroom problems.

5.4 Problem Formulation as an Integer Linear Programming Problem (ILP)

In this section we present a detailed discussion of the formulation of the ILP model, the known (given) input, the variables and the output. The ILP formulation code is given in Appendix C.

5.4.1 Input

The inputs of the problem can be listed as follows:

1. n = number of nodes in the network, i.e. $|\mathbf{V}|$.
2. s = maximum number of services requested.
3. p = maximum number of useful paths in the network.
4. e = total number of links/edges in the network, i.e. $|\mathbf{E}|$.
5. Path-link matrix **L** is a binary p x e dimensional that indicates whether a link is used in a path or not.
6. Bandwidth vector $\boldsymbol{B}$ is e dimensional that gives capacity of a link/edge and $b_{\max}$ is maximum link capacity (in the entire network), where b_l is the bandwidth of link l.
7. Service installation cost matrix **C**, is n x s dimensional that quantifies the cost for installing a service on a node.
8. Discount Γ is given if multiple services are installed on node.

9. A three dimensional binary routing matrix, **R** that indicates the path used between a source-destination pair in the network.
10. A throughput demand matrix **T** that gives the throughput required for a service at a consumer.
11. A delay demand matrix **D** that is the maximum delay allowed for a service at a consumer.
12. A load-delay lookup table **Q** that approximates the delay, due to queuing, transmission and propagation, on a link given the load on the link, $q_{\max}$ is maximum load lookup and q is the interval of load values in **Q**.

5.4.2 Variables and their definitions

The variables for the problem can be defined as follows:

1. A binary service location matrix **X**, is n x s dimensional, that indicates whether a service is installed on a node or not.
2. A path-service capacity matrix, **Z** is p x s dimensional, that quantifies the capacity of a service on a path.
3. An n dimensional service installation indicator vector $\boldsymbol{U}$ that indicates whether multiple services are installed on a node.
4. A binary service-path indicator (normalized **Z**) matrix **Y** is p x s, and indicates if a service uses a path or not.
5. An e dimensional link load vector **V** that quantifies the load on a link.
6. An e dimensional link-delay vector **G** that gives the delay on a link.

7. A p dimensional path delay vector **H**, gives the total end-to-end delay on a path.
8. An indicator variable r is n x n x s x p, such that

$$r_{i,j,k,m} = \begin{cases} 1, & \text{if } i \text{ provides service } k \text{ to } j \text{ via path } m \\ 0, & otherwise \end{cases}$$

Variable **X** is also the output of the problem since it gives the optimal service installation configuration.

5.4.3 The ILP Formulation

Based on the inputs and variables we formulate the ILP model as follows, with constraints in canonical form, for easy implementation during Lagrangean Relaxation.

Minimize the cost of service installation

$$\min \quad \left\{ \sum_{i=1}^{n} \sum_{k=1}^{s} C_{i,k} \cdot X_{i,k} + \sum_{i=1}^{n} \Gamma \cdot U_i \right\}$$

subject to:

Service installation cost and location constraints:

1) $\sum_{k=1}^{s} (X_{i,k}) - p \cdot b_{\max} \cdot U_i \leq 0$, $\qquad \forall 1 \leq i \leq n$

2) $-\sum_{k=1}^{s} (X_{i,k}) + U_i \leq 0$, $\qquad \forall 1 \leq i \leq n$

Throughput Constraints

3) $-\sum_{i=1}^{n} \sum_{m=1}^{p} R_{i,j,m} \cdot Z_{m,k} \leq -T_{j,k}$, $\qquad \forall 1 \leq i \leq n, 1 \leq k \leq s$

4) $X_{i,k} - \sum_{j=1}^{n}\sum_{m=1}^{p} R_{i,j,m} \cdot Z_{m,k} \le 0,$ $\forall 1 \le i \le n, 1 \le k \le s$

5) $-p \cdot b_{\max} \cdot X_{i,k} + \sum_{j=1}^{n}\sum_{m=1}^{p} R_{i,j,m} \cdot Z_{m,k} \le 0,$ $\forall 1 \le i \le n, 1 \le k \le s$

<u>Network Link Capacity Constraint</u>

6) $\sum_{m=1}^{p}\sum_{k=1}^{s} L_{m,l} \cdot Z_{m,k} \le B_l,$ $\forall 1 \le l \le e$

<u>Delay Constraints:</u>

7) $-p \cdot b_{\max} \cdot Y_{m,k} + Z_{m,k} \le 0,$ $\forall 1 \le m \le p, 1 \le k \le s$

8) $Y_{m,k} - Z_{m,k} \le 0,$ $\forall 1 \le m \le p, 1 \le k \le s$

rounding load on links to match up with lookup table:

9) $V_l - \sum_{m=1}^{p}\sum_{k=1}^{s} \left(L_{m,l} \cdot Z_{m,k}\right) \le 0,$ $\forall 1 \le l \le e$

10) $-V_l + \sum_{m=1}^{p}\sum_{k=1}^{s} \left(L_{m,l} \cdot Z_{m,k}\right) \le 0,$ $\forall 1 \le l \le e$

11) $V_l - q \cdot a_l \le 0,$ $\forall 1 \le l \le e$

12) $-V_l + q \cdot a_l \le 0,$ $\forall 1 \le l \le e$

finding index to be looked up in the delay table:

13) $V_l - q \cdot i - \left(b_{\max} + q_{\max}\right) \cdot c_{l,i} \le 0,$ $\forall 1 \le l \le e, 1 \le i \le \frac{b_l}{q} + 1$

14) $-V_l + q \cdot i - \left(b_{\max} + q_{\max}\right) \cdot c_{l,i} \le 0,$ $\forall 1 \le l \le e, 1 \le i \le \frac{b_l}{q} + 1$

15) $\sum_{i=1}^{|b_l|} c_{l,i} \le \frac{b_l}{q}$ $\qquad \forall 1 \le l \le e$

16) $k_{l,i} + c_{l,i} \le 1$, $\qquad \forall 1 \le l \le e, 1 \le i \le \frac{b_l}{q} + 1$

17) $-k_{l,i} - c_{l,i} \le 1$, $\qquad \forall 1 \le l \le e, 1 \le i \le \frac{b_l}{q} + 1$

looking up delay in lookup table and computing delay on link and path:

18) $G_l - \sum_{i=1}^{|Q|} (k_{l,i} \cdot Q_i) \le 0$, $\qquad \forall 1 \le l \le e$

19) $-G_l + \sum_{i=1}^{|Q|} (k_{l,i} \cdot Q_i) \le 0$, $\qquad \forall 1 \le l \le e$

20) $H_m - \sum_{l=1}^{e} (L_{m,l} \cdot G_l) \le 0$, $\qquad \forall 1 \le m \le p$

21) $-H_m + \sum_{l=1}^{e} (L_{m,l} \cdot G_l) \le 0$, $\qquad \forall 1 \le m \le p$

meeting delay requirement:

22) $-R_{i,j,m} \cdot Y_{m,k} - r_{i,j,k,m} \le -1$, $\qquad \forall 1 \le i, j \le n, 1 \le k \le s, 1 \le m \le p$

23) $R_{i,j,m} \cdot Y_{m,k} + r_{i,j,k,m} \le 1$ $\qquad \forall 1 \le i, j \le n, 1 \le k \le s, 1 \le m \le p$

24) $H_m - q_{\max} \cdot r_{i,j,k,m} \le D_{j,k}$ $\qquad \forall 1 \le i, j \le n, 1 \le k \le s, 1 \le m \le p$

to make sure only those paths offer a service capacity that are between a producer and a consumer:

25) $-q_{\max} \cdot p \sum_{i=1}^{n} \sum_{m=1}^{p} R_{i,j}^{m} Y_{m,k} \le -T_{j,k}$ $\qquad \forall 1 \le j \le n, 1 \le k \le s$

$$26)\sum_{i=1}^{n}\sum_{m=1}^{p} R_{i,j}^{m} Y_{m,k} \leq T_{j,k} \qquad \forall 1 \leq j \leq n, 1 \leq k \leq s$$

5.5 Discussion of the ILP Formulation

Here we interpret the constraints presented in the above mathematical formulations [40]. The constraints relate to; service installation and location, meeting throughput, factoring underlying network link layer capacity, *discretization* of load, approximating delay for load by performing a lookup in a load-delay table, computing end-to-end delay on a path and meeting delay requirements.

In the service location problem, our goal is to install service(s) on nodes in the network that meet throughput and delay requirements for requests, while minimizing the service installation costs and promoting service federation. Therefore, the objective function is to minimize the service installation cost and give a discount for multiple service installations, captured by the service installation indicator vector **U**. We capture service installation location details in constraints (1) and (2) where $X_{i,k} = \begin{cases} 1, \text{if service } k \text{ is installed on } i \\ 0, otherwise \end{cases}$.

Constraint (3) ensures that the sum of service capacity provided across all paths leading to destination j (consumer j) with throughput demand of $T_{j,k}$ for service k is met. Constraint (4) and (5) ensure that the paths providing the service capacity come from nodes where the service is installed, that is from a service provider i.

Constraint (6) captures the network link layer capacity and restricts the service capacity on a path so that it does not exceed the underlying individual link capacity of the links in a path. Constraint (7) and (8) compute the service-path indicator, a binary variable $Y_{m,k} = \begin{cases} 1, \text{if } Z_{m,k} \geq 1 \\ 0, otherwise \end{cases}$.

The first six constraints allow us to meet throughput requirements while minimizing service installation costs. However, our SLP problem meets QoS requirements of throughput and delay. Therefore, we next discuss the constraints to meet the delay requirements.

Constraints (9) and (10) compute the load on a link as the sum of service capacity across all paths using the link and are equivalent to $V_l = \sum_{m=1}^{p} \sum_{k=1}^{s} \left(L_{m,l} \cdot Z_{m,k} \right)$. Constraint (11) ensures that load on the link, V_l is defined in the load-delay lookup table Q with load interval q, with $a_l = \frac{V_l}{q}$ (equivalent to (11 and 12)). Constraints (13), (14) check whether there is a difference between load on a link and the respective lookup load in the table for a given link. Constraint 15 ensures that there is at least one entry in the lookup table that matches the load on the link, indicated by 0. Constraints (16) and (17) compute $1-c_{l,i}$, so that the lookup can be performed, which will be achieved by multiplying $k_{l,i}$ with Q_i in the respective row that matches the maximum bandwidth of the link. Constraints (18) and (19) compute the delay on a link G_l as $G_l = \sum_{i=1}^{|Q|} \left(k_{l,i} \cdot Q_i \right)$.

End-to-end delay of a path is computed as the sum of the delay across all links in the path, that is, $H_m = \sum_{l=1}^{e} (L_{m,l} \cdot G_l)$ (equivalent to (20) and (21)). Constraints (22), (23) and (24) ensure that the end-to-end delay of a path does not exceed the delay requirement $D_{j,k}$. That is, every path between a service provider *i* and consumer *j* for service *k* must not exceed delay requirement of consumer for service *k*, equivalent to $R_{i,j}^{m} \cdot Y_{m,k} \cdot H_m \leq D_{j,k}$. However, this is nonlinear, therefore, constraints (22) and (23) compute a binary variable $r_{i,j,k,m} = \begin{cases} 1, \text{ if } R_{i,j}^{m} Y_{m,k} = 0 \\ 0, otherwise \end{cases}$, so if $r_{i,j,k,m} = 1$, then the path delay and delay requirement comparison is not valid, however, if $r_{i,j,k,m} = 0$, then we want to ensure that the path delay does not exceed delay requirement, this is captured in constraint (24). Two safety constraints, constraints (25) and (26) are added to ensure that only those paths carry services or offer services that lead from a provider to a consumer.

5.6 Service Location and Planning in Oppnets

Let us consider a home equipped with an Oppnet-enabled laptop, webcam and a smart fire detector, as illustrated in Fig. 9. Now, assume that you were not home and were concerned about a fire or robbery in your home. In the case of a fire scare, it would be beneficial if you could initiate a connection between your cell phone (that you are carrying) and your laptop at home through the wireless Internet infrastructure of your cellular service provider. Once initiated, the laptop can detect and connect with the webcam and smart fire detector, through the Bluetooth and Internet of the Home Area Network (HAN), respectively.

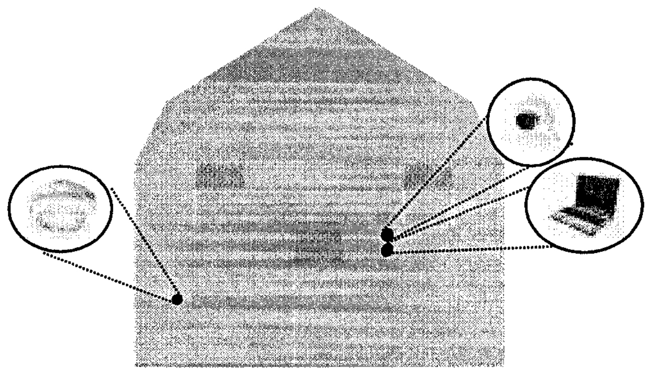


Figure 9. A home equipped with an Oppnet-enabled laptop

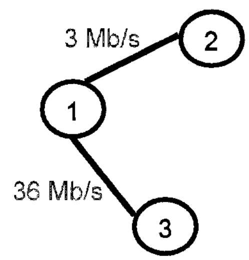
Now, the laptop (in this case the seed) in the Oppnet can process images captured by the webcam (in this case, this is an Oppnet helper) and interpret the smoke concentration levels recorded by the fire detector, which is also an Oppnet helper, and inform you that there is no fire, based on the smoke concentration levels and the image captured by the webcam that shows everything to be normal.

The Service Location-Planning (SLP) technique can help realize this theoretical Oppnet. In this section, we will illustrate how SLP can be used in Oppnets.

Firstly, an Oppnet is configured, illustrated in Fig. 10(a), starting with the seed, the Oppnet-enabled laptop (labeled as node 1 in Fig. 10(a)), which detects and incorporates the webcam (node 2) and smart fire detector (node 3), as Oppnet helpers using Oppnet primitives (cf. Appendix A). The data rates of 3 Mb/s and 36 Mb/s illustrated in Fig. 10(a) are typical data exchange rates in the Bluetooth and Internet medium.

Then the seed can probe the Oppnet helpers for their resource capabilities and construct a table of Service Installation Costs, presented in Fig 10(b). Let us assume that the Oppnet can offer only four services– capturing images, collecting smoke concentration levels, communication and processing, depicted as Service A, B, C and D in Fig. 10(b), respectively. Figure 10(c) captures the fact that the laptop needs to 2 units of Service A, i.e. capture images, and 20 units of Service B, i.e. collecting smoke concentration levels., with no later than 10 seconds of delay.

These input parameters passed to the ILP model of the SLP problem, yields the output presented in Fig. 10(d), concurring with our theoretical output, that is use Images captured by the webcam, i.e. install Service A on Node 2, and collect smoke concentration levels from the smart fire detector, i.e. install Service B on Node 3. Note, in this scenario, Service A and Service B do not need to be installed on Nodes 2 and 3, they can just be used; this is also captured in the lower Service Installation Costs of Service A on Node 2 and Service B on Node 3 and very high Service Installation Cost of Service A and B on Node 1.



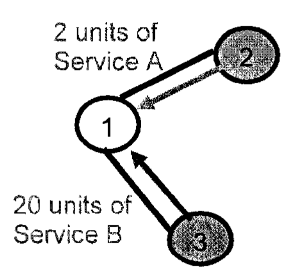
(a) A small Oppnet

Node	Service A	Service B	Service C	Service D
1	5000	5000	10	10
2	20	5000	5000	5000
3	5000	40	5000	5000

(b) Input parameter: table of service installation costs for 2 services for the above topology

Node	Throughput Requirement (units)		Delay Requirement (s)	
	Service A	Service B	Service A	Service B
1	2	20	10	10
2	0	0	0	0
3	0	0	0	0

(c) The input parameters, of the SLP problem



(d) The output/solution to the above SLP problem

Figure 10. Resource utilization in Oppnets using Service Location-Planning technique

In this way the Service Location-Planning technique enables resource utilization in small-scale Oppnets.

5.7 Service Location and Planning in Generic Networks

We used lp_solve [39] as the ILP solve engine that uses the simplex method to solve integer linear programs. All our test scenarios are based on the topology of a carrier's nationwide IP backbone network topology illustrated in Fig. 11. For small-scale networks, we can abstract smaller topologies from this large-scale network. For example, a six-node network can be abstracted from this topology, highlighted in yellow in Fig. 11 and renumbered and illustrated in Fig. 12(a).

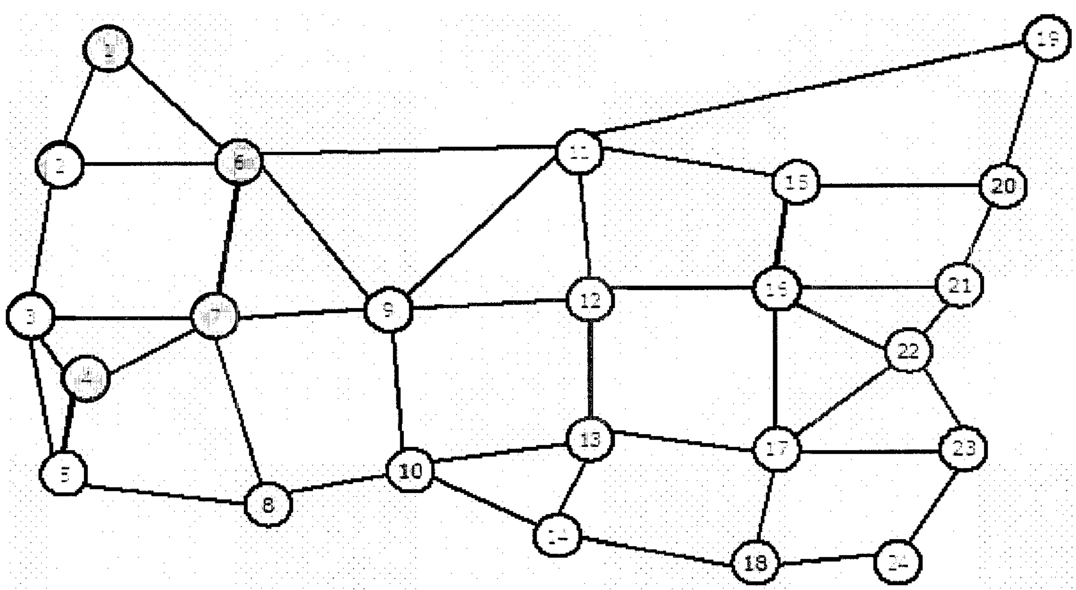
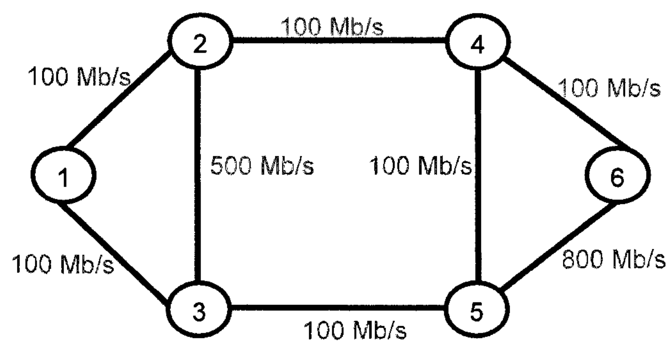


Figure 11. A carrier's nation-wide IP backbone network topology [48]

In Fig. 12 we use this generic small-scale network to illustrate how our ILP formulation of the SLP problem optimally installs services in this generic network to meet service requests and QoS parameters. The input for the ILP formulation of the small-scale network is shown in Fig. 12(b) and 12(c), the output for the problem is given in Fig. 12(d).



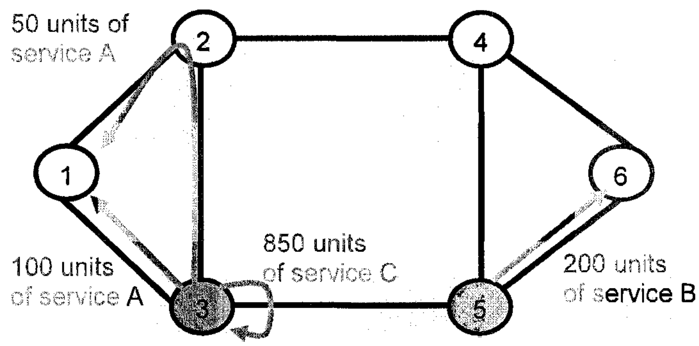
(a) A sample 6-node topology scenario (also appears in [48])

Node	Service A	Service B	Service C
1	110	100	100
2	120	100	150
3	100	100	80
4	100	120	120
5	100	100	110
6	100	110	90

(b) Input parameter: table of service installation costs for 3 services for the above topology

Node	Throughput Requirement (units)			Delay Requirement (s)		
	Service A	Service B	Service C	Service A	Service B	Service C
1	137	0	0	1000	0	0
6	0	180	0	0	1000	0
3	0	0	850	0	0	100

(c) The input parameters, of the SLP problem



(d) The output/solution to the above SLP problem

Figure 12. Resource utilization in generic networks using Service Location-Planning technique

In this scenario, as Fig. 12(d) shows, it is optimal to install Service A on Node 3 and Service B on Node 5. Furthermore, service splitting occurs since all link bandwidths are 100 Mb/s, so 100 units of Service A are provided via the direct path (3,1) and the indirect path (3,2,1) provides 50 units of Service A (as indicated with arrows in Fig. 12(d)). The delay for both of these paths meets the delay requirements of 1000 seconds since the direct path has a delay of 500 seconds, and the indirect path has a total delay of 400 seconds (200 seconds delay on link 2–3 and 200 seconds on link 2–1). Also note that, the service provided exceeds the throughput requested of 137 units by 13 units, since the delay lookup table in Fig. 8 was used, which is enumerated for loads of increments of 50 units. Thus, instead of providing 37 units on the indirect path, 50 units are provided. It is true that some bandwidth is wasted however the tradeoff is linear computation of delay.

Similarly, 200 units of Service B are provided from Node 5 directly to Node 6, 200 units via the direct path (5,6) with a delay of 16 seconds, which is less than the consumer-defined delay requirements of no more than 1000 seconds.

We also meet the request of Node 3 for 850 units of Service C with a delay of no more than 100 seconds. Firstly, note that Service C is cheapest from Node 3. Secondly, since node 3 already has a previous service installed on it, i.e. Service A, therefore installation of Service C, another service on Node 3, entitles it to a discount of 20. Without this discount, the cost of installing Service A and C on Node 3 would have been, 220. However, with the service federation discount, the cost of installing Service A and C on Node 3 is 200. Thus it is cost-effective to install Service C on Node 3. Hence, 850 units of Service C is provided to Node 3 locally, through the zero-cycle loops, with a delay of 67 seconds.

In this manner, the ILP formulation achieves optimal service installation costs for a small-scale network. However, for the SLP problem discussed in this paper, it is obvious that there are various levels of complexity. The dimensions of complexity in a SLP can be delineated in terms of:

- number of nodes, n,
- number of services, s,
- number of paths, p,
- links/edges, e,
- bandwidth capacity, b_l (cf. ILP formulation)

- lookup table interval, q (defined in 5.4.1).

Increasing any of these dimensions causes the problem to grow tremendously. For example, for a problem with 6 nodes, 2 services and 22 paths, the number of variables is 480, where as a problem with 6 nodes, 3 services and 22 paths has 538 variables. This is an increase of 58 variables for just an increase by one service in the SLP problem.

The numbers of variables for different scenarios of the SLP problem are presented in Table 5 as *x*n-*y*s-*z*p, where x, y and z are the number of nodes, services and paths, respectively. If two scenarios have the same label then the difference is in the bandwidth capacities of the underlying links. We keep the number of edges (e) constant across scenarios with the same n, s, p dimensions. Also, the same lookup table is used in all scenarios.

Table 5

Scenarios and Number of Variables

Scenario	Number of variables in ILP
6n-2s-22p	480
6n-3s-22p	538
6n-3s-22p	598
10n-2s-26p	762
10n-2s-26p	826
15n-2s-66p	1239
20n-3s-82p	1896
24n-2s-58p	1907
24n-2s-58p	1937
24n-2s-32p	1910

In Fig. 13 and 14 we illustrate the growth in the number of constraints in an ILP and the execution times with respect to the number of variables in a SLP problem. The execution times (in seconds) are computed based on the Windows XP platform running on a Pentium IV, 3.0 GHz laptop with 384 MB of RAM.

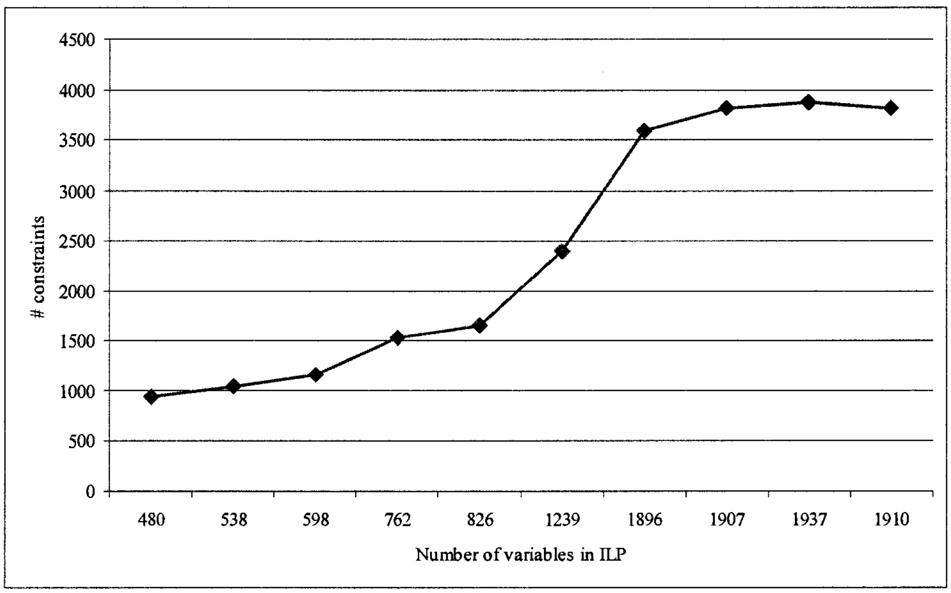


Figure 13. Number of constraints vs. number of variables in an ILP

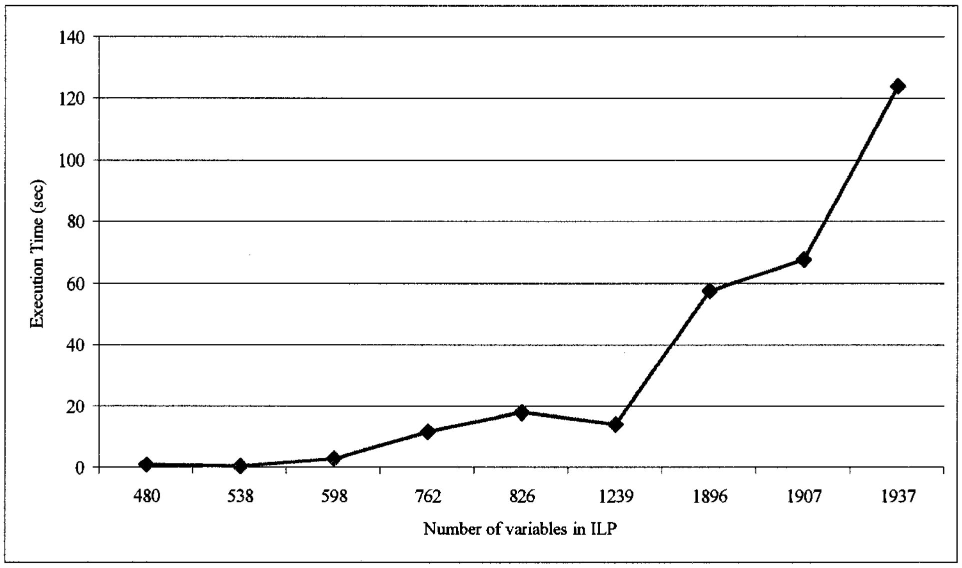


Figure 14. Execution times vs. number of variables in ILP

5.8 Flexibility of Service Location and Planning (SLP) Problem

It should be evident from the formulation and the test scenarios and results, that the solution of the SLP is generic enough to be adapted for any network.

For example, consider the service location and planning problem in terms of Cisco's AON technology that enables relocation of application services from end nodes (end-points) in the network to the routers and switches in a network [8]. This is a huge-impact technology since it allows for application-level message intelligence at the router level.

AON has two obvious fundamental consequences. First, AON routers can route application-layer messages, such as stock quotes, weather and news alerts, etc., to the appropriate application service rather than to an arbitrary IP address. Second, services can be installed on AON routers. This enables faster satisfaction of service demand requests, as these requests are not propagated to the end points of the network, rather interpreted and satisfied within the network (at the router/switch level).

Consider an AON with *n* nodes in a network requesting *s* services with pre-determined throughout and delay parameters. The goal is to install services within the network to meet the requests and satisfy the QoS parameters, while minimizing service installation costs. In such scenarios, the Service Location and Planning (SLP) problem can be solved to minimize service installation costs and meet consumer's requests.

CHAPTER 6

SLP PROBLEM FOR LARGE-SCALE NETWORKS

6.1 An Introduction to Lagrangean Relaxation

Hard or complex optimization problems can be simplified or approximated by making various relaxations or approximations, for example, Linear Programming (LP) relaxation, Lagrange Relaxation, etc. With LP relaxation, the integer constraints of a linear program are ignored. However, generally the solution to an LP relaxed problem does not fulfill all the discrete requirements [41].

In constrained optimization problems, LR proves particularly useful for separable nonlinear programming problems or for integer linear programming problems [43]. *Separable nonlinear programming problems* are those that contain some linear parameters/constraints, and removing them from the problem leaves a problem involving only nonlinear parameters [44]. *Integer Linear Programming (ILP)* problems, are those that consist of linear functions/constraints and integer variables.

A useful observation of hard problems is that they can often be viewed as easy problems complicated by a relatively small set of side constraints [45]. By "dualizing" the complicating constraints, a simpler (compared to the original problem), easy to solve Lagrangean problem is obtained [45].

Thus, we are interested in *Lagrangean Relaxation of ILP* problems, where multipliers, also known as Lagrangean Multipliers, are attached to the complicating constraints and moved into the objective function [44]. In this way, the multipliers are used to control the Lagrangean dual such that when a solution to the problem violates the complicating constraint, the multipliers are used to penalize the objective function for ILP. The optimal value of the Lagrangean dual ILP objective function gives a lower bound for minimization problems and an upper bound for maximization problems, if multipliers are always positive.

6.1.1 Illustration of Lagrangean Relaxation technique

Let's briefly illustrate the Lagrangean Relaxation technique. Consider for example, an ILP minimization problem L defined as follows.

$$L = \min fx$$

subject to:

$$Ax \leq b \qquad (1)$$

$$Cx = d \qquad (2)$$

$$x = \{0,1\} \qquad (3)$$

Where, x is the problem variable and f, A and C are problem dependent input parameters and b and d are constants. Note, that the third constraint makes this a combinatorial problem.

Let us assume that the first constraint is the complicating constraint.

A straight-forward relaxation approach would be to remove the complicating constraint (1) and solve the problem without the complicating constraint and if the solution satisfies the complicating constraint then it is optimal for the original problem

[52]. However, using the Lagrangean Relaxation for ILP problems, we formulate a Lagrangean dual L^* that is defined below and takes into account the complicating constraint(s).

$$L^* = \min(fx + \lambda_1(Ax - b))$$

subject to:

$$Cx = d \qquad (2)$$

$$x = \{0,1\} \qquad (3)$$

Where, λ_1 is the Lagrangean multiplier.

Therefore, for *a valid solution for x and for $\lambda_1 \geq 0$,*

$\Rightarrow Ax \leq b$, since for valid solution x, constraint (1) must be met, i.e. $Ax \leq b$

$\Rightarrow Ax - b \leq 0$

$\Rightarrow \lambda_1(Ax - b) \leq 0$, given $\lambda_1 \geq 0$

$\Rightarrow L^* \leq L$, Note: $L^* = L$ is not guaranteed

This is how, the Lagrangean multipliers are used to control the solution x and penalize the objective function when a solution to x violates the complicating constraint(s).

Again, note that the Lagrange relaxation dual forms a lower bound for minimization problems, provided that the multipliers are not negative. In other words, we try to find the highest lower bound of the Lagrangean dual for close-to-optimal solution of the original problem. However, if the multipliers are allowed to be negative, then $L^* \geq L$ or $L^* \leq L$ could still hold true, in the case where the negative multipliers are small in magnitude. Thus, if the multipliers are not bounded to be positive the Lagrangean dual will not produce a bound for the problem, instead it will just converge to a valid solution irrespective of the cost.

Nonetheless, goal of the Lagrange relaxation technique is to iteratively update the Lagrange multipliers through different algorithms, such as subgradient, surrogate gradient [43], [44], a technique we refer to as the *CMU technique* (Carnegie Mellon University technique) [46], etc. However, irrespective of the technique used, the multipliers are updated such that they represent the cost of violating the constraints they represent. Therefore, we only briefly review some of the Lagrangean Relaxation techniques, namely, the subgradient, surrogate gradient and the CMU techniques.

6.1.2 Different Lagrange Relaxation Techniques

a) Subgradient and Surrogate Gradient Methods for Updating Lagrange Multipliers

For simplicity mathematical proofs and theorem are omitted and we briefly discuss the intuition behind subgradient and surrogate gradient methods and refer readers to [43] for a more detailed discussion.

The Lagrangean dual is nondifferentiable at an optimal point, but subdifferentiable everywhere else [45]. Thus, the subgradient method is an adaptation of the gradient method in which the gradient is substituted with the subgradient [3]. In the subgradient method, the multipliers are updated according to the equation $\lambda^{k+1} = \lambda^k + s^k g^k$, where λ^k – is the multiplier, s^k is the step-size and g^k is the subgradient of the dual Lagrangean function at iteration k. To compute the subgradient requires a solution of all the subproblems [43]. The step-size is an application dependent variable and can be computed according to a mathematical formula. However, an easier rule that has performed well empirically is to set s^k to 2 and halved whenever L^* fails to increase after some fixed number of iterations [45].

In an effort to reduce the time consumption of the subgradient method, where all subproblems have to be solved to get the subgradient at an iteration, the surrogate gradient method updates the multipliers in a similar manner to the subgradient method, however, does not require a solution to all subproblems, hence reducing the time consumption at every iteration and reducing the total time for optimizing the Lagrangean dual.

b) CMU Technique for Updating Lagrange Multipliers

In the multiplier update method of [46], which we termed CMU (Carnegie Mellon University) method, time consumption is shortened significantly as the need to compute any subgradient or surrogate gradients is eliminated and a very straight forward approach is used to update the multipliers. We adapted this algorithm for minimization problems and present it in Fig. 15. In short, the multipliers are updated by a constant k, in each iteration based on whether they violate or give slack to a constraint.

For a binary linear minimization problems and with all constraints in canonical form

1. Begin with each λ at 0, with step size k (problem dependent value)
2. Solve the Lagrangean dual to get current solution x.
3. For every constraint violated by x, increase corresponding λ by k.
4. For every constraint with positive slack relative to x, decrease the corresponding λ by k.
5. If m iterations have passed since the best relaxation value has increased, cut k in half.
6. Go to 2.

Figure 15. CMU Lagrange Relaxation Technique [46] adapted for binary minimization problems

Provided all these different techniques for implementing Lagrange relaxation, two properties are important in evaluating which relaxation technique to use, the sharpness of the bounds produced and the amount of computation time required to obtain these bounds [44]. Usually, there is a tradeoff between these two properties [44].

Our goal is simplicity, in terms of computationally intensity, thus we adapt the CMU Lagrangean Relaxation technique to extend the SLP model for large-scale networks.

Scrutiny of the CMU technique yields the fact that this Lagrange relaxation technique only works for *binary* linear integer programming problems. Next, we illustrate this technique on a small binary minimization optimization problem and later discuss how this technique can be adapted to our *integer* linear service location and planning problem, which is not a binary problem.

6.2 An Illustrative Example for Lagrangean Relaxation Using the Adapted CMU Technique

In this section, we demonstrate the CMU Lagrangean Relaxation technique that has been adapted for binary linear minimization problems (cf. Fig. 15).

Consider the simple minimization problem defined below in its canonical form.

$$Z = \min \qquad 4x_1 + 5x_2 + 6x_3 + 7x_4$$

subject to

$$2x_1 + 2x_2 + 3x_3 + 4x_4 \leq 7 \qquad (1)$$

$$x_1 - x_2 + x_3 - x_4 \leq 0 \qquad (2)$$

$$-x_1 - x_2 \leq -2 \qquad (3)$$

$$x_j \in \{0,1\}, \forall 1 \leq j \leq 4$$

The optimal solution to this minimization problem is 9, with $x_1 = x_2 = 1$. Below, various iterations of the CMU technique are illustrated for exemplification of the Lagrangean Relaxation of this small binary linear minimization problem. If all constraints are to be relaxed, that is, consider all the constraints as complicating constraints, then the Lagrangean dual would be defined as follows.

$$Z^{*} = \min \begin{Bmatrix} 4x_1 + 5x_2 + 6x_3 + 7x_4 \\ + \lambda_1(\text{constraint1}) \\ + \lambda_2(\text{constraint2}) \\ + \lambda_3(\text{constraint3}) \end{Bmatrix}$$

subject to $\quad x_j \in \{0,1\}, \forall 1 \leq j \leq 4$

$$\Rightarrow \qquad Z^{*} = \min \begin{Bmatrix} 4x_1 + 5x_2 + 6x_3 + 7x_4 \\ + \lambda_1(2x_1 + 2x_2 + 3x_3 + 4x_4 - 7) \\ + \lambda_2(x_1 - x_2 + x_3 - x_4 - 0) \\ + \lambda_3(-x_1 - x_2 + 2) \end{Bmatrix}$$

subject to $\quad x_j \in \{0,1\}, \forall 1 \leq j \leq 4$

Now, given, step size $k = 0.5$ and $m = 5$ and Z^* as above.

In iteration 1, all $\lambda_i = 0$, $1 \le i \le 3$,

$$\Rightarrow \min\begin{Bmatrix} 4x_1 + 5x_2 + 6x_3 + 7x_4 \\ +0(2x_1 + 2x_2 + 3x_3 + 4x_4 - 7) \\ +0(x_1 - x_2 + x_3 - x_4 - 0) \\ +0(-x_1 - x_2 + 2) \end{Bmatrix} = \min\ 4x_1 + 5x_2 + 6x_3 + 7x_4$$

Thus, the bound is 0, with all $x_j = 0$, $1 \le j \le 4$. This solution gives positive slack to constraint (1), just meets constraint (2), and violates constraint (3), thus the multiplier are updated and are $\lambda_1 = -0.5$, $\lambda_2 = 0$, and $\lambda_3 = 0.5$.

In iteration 2, the Lagrangean dual becomes

$$\min\begin{Bmatrix} 4x_1 + 5x_2 + 6x_3 + 7x_4 \\ -0.5(2x_1 + 2x_2 + 3x_3 + 4x_4 - 7) \\ +0.5(-x_1 - x_2 + 2) \end{Bmatrix} = \min\ 2.5x_1 + 3.5x_2 + 4.5x_3 + 5x_4 + 4.5$$

The optimal solution to this dual is $x_j = 0$, $1 \le j \le 4$ with bound = 4.5 and multipliers are updated to $\lambda_1 = -1$, $\lambda_2 = 0$ and $\lambda_3 = 1$. Since the current solution to the LR dual gives positive slack to constraint (1), violates constraint (3) and still meets constraint (2).

In this manner, we play with the multipliers until iteration 14 when the bound, the value of the LR dual, has failed to increase in 5 ($m = 5$) iterations. In this case, we cut the step size, k, of 0.5 in half, to get $k = 0.25$ and at which point the multipliers are $\lambda_1 = -2$, $\lambda_2 = -0.5$ and $\lambda_3 = 3$. And we continue to update the multipliers as demonstrated in iterations 1 and 2, above.

In iteration 16, we find a higher bound of 14.25, and in iteration 21, when the bound fails to increase any further, we cut the step size in half, yielding the new step size of $k = 0.125$.

We continue in this manner to iteration 23, where the bound of 14.5 is achieved, and when the bound fails to increase in the next 5 iterations, in iteration 28 we cut the step size to 0.0625. And in iteration 33, the bound doesn't increase any further and again we cut the step size to half, to get $k = 0.03125$.

In our program, our termination criterion is $k < 0.05$. Thus, we terminate the Lagrange relaxation technique in 33 iterations. At which point, the solution is x_1=1, x_2=1, x_3=x_4=0 and the bound=14.5.

This solution set is the optimal solution set, and when substituted into the original objective function (Z) we get the optimal objective value of 9. The Lagrange multipliers at this termination stage are $\lambda_1 = -1.875$, $\lambda_2 = 0.4375$, and $\lambda_3 = 3$.

In this manner, we get the optimal solution of $x_1 = 1$, $x_2 = 1$, $x_3 = x_4 = 0$ with a bound=14.5 (the LR dual value), which is higher than the optimal objective function value. In this case, the LR bound did not provide a lower bound of the original problem; this is because we allowed the multipliers to be negative.

Furthermore, scrutiny of this simple CMU technique yields the fact that this Lagrange relaxation technique only works for binary linear integer programming problems. Therefore, we will further adapt it for our SLP problem that is an Integer Linear Programming (ILP) problem.

6.3 Lagrangean Relaxation of Service Location and Planning Problem

As discussed earlier, ILP models can be solved for small-scale networks efficiently. For larger-scale networks relaxations techniques, approximation techniques and other heuristics and meta-heuristics have to be used. They may yield suboptimal results, but obtain solutions for larger-scale networks, which cannot be handled by LP solve engines (*solvers*).

Recall that Lagrange relaxation techniques are used to remove/relax complicating constraints from the ILP model, making the model simpler, and associating a multiplier with the relaxed constraint (the removed constraint) so that the cost of violating or removing the constraint is captured by the multiplier.

Also recall that we wanted to use the simple and straight-forward approach of the CMU technique. We adapted the CMU technique for binary linear minimization problems in Fig. 15. However, the technique is applicable only to binary linear minimization problems, and our SLP problem is a non-binary integer linear minimization problem. Therefore, we only relax/remove those constraints that are constituted of binary variables, such as constraints (1), (2) and (17) in the ILP formulation of Chapter 5.4.3. We will show significant reduction in the number of constraints in the Lagrangean dual of the SLP problem even with relaxing just the above mentioned three constraints.

The Lagrangean Relaxation code is presented in Appendix D and sample input and output files for a sample Lagrangean Relaxation (LR) scenario is presented in Appendix E.

6.4 Test Scenarios and Results

Using the Lagrangean relaxation technique we compare the reduction in the number of constraints in a SLP problem in Fig. 16 also tabulated in Table 6. This illustrates the improvement with Lagrangean relaxation over the ILP formulation of the SLP problem. Fig. 16 illustrates a reduction of 15 – 18% (at the least 15.6% and at the most 18.8%) of the total constraints, that is, an average improvement of 17.42% (cf. Table 6). This yields reductions of upto 729 constraints in a 24 node, 2 service and 58 path network setup. We reduce the complexity of the problem by only utilizing useful alternate paths (with a minimum number of edges traversed), rather than all possible paths between all source and destination pairs, which in itself is a hard problem.

The scenarios delineated in Fig. 16 and Table 6 as *x*n-*y*s-*z*p, where *x*, *y* and *z* are the number of nodes, services and paths respectively. If two scenarios are labeled the same, then the difference is the bandwidth capacity of an underlying link(s). We keep the number of edges (*e*) constant across scenarios with same *n*, *s*, *p* dimensions. Also, the same lookup table is used in all scenarios.

Table 6 shows how Lagrangean relaxation bounds the ILP objective function value. It should be noted that the Lagrangean dual does not always represent a lower bound on the optimal solution (ILP cost) for all scenarios since we allowed multipliers to be negative. These results can be further increased by relaxing more binary constraints from the SLP model in the Lagrangean Relaxation.

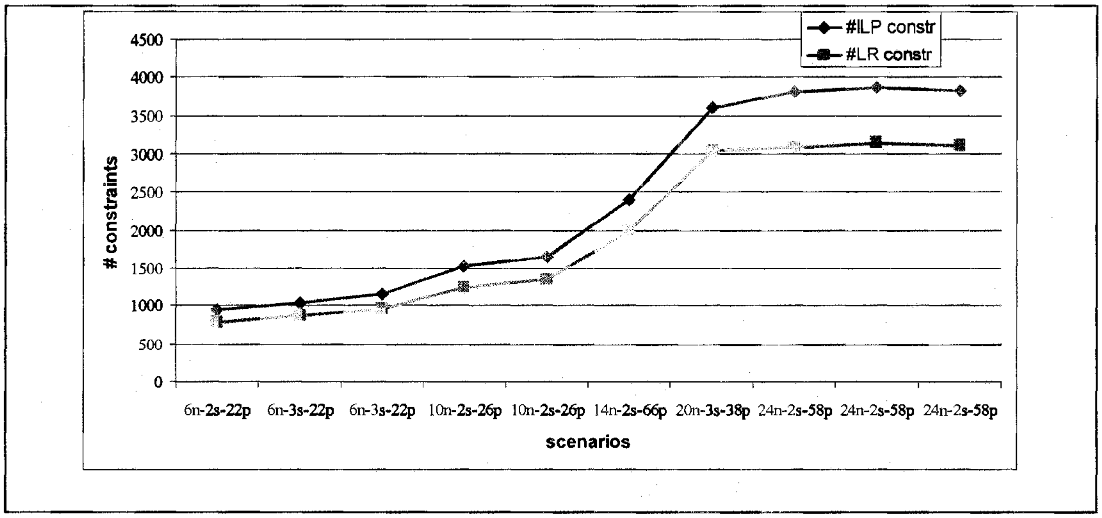


Figure 16. Comparison of number of constraints with ILP vs. LR of SLP problem

Table 6

ILP Cost and LR Lower Bound Comparison

Scenario	ILP cost	LR lower bound	# constraints in ILP	# constraints in LR	% improvement in # constraints
6n-2s-22p	240	219.62	940	778	17.23
6n-3s-22p	340	324.93	1038	876	15.61
6n-3s-22p	340	298	1158	966	16.58
10n-2s-26p	240	220	1528	1250	18.19
10n-2s-26p	180	168	1656	1346	18.72
15n-2s-66p	60	48	2395	2001	16.45
20n-3s-82p	100	68	3594	3050	15.14
24n-2s-58p	160	236	3810	3095	18.77
24n-2s-58p	280	320	3872	3143	18.83
24n-2s-32p	160	220	3819	3104	18.72

Average improvement = 17.42

LR = Lagrangean Relaxation

CHAPTER 7

CONCLUSION

7.1 Summary

The new computing paradigm discussed in this Thesis, termed Class 2 Opportunistic Networks (Oppnets) can be considered to be the next level in the continuously evolving computing paradigms that are trying to achieve the demand for seamless, untethered, ubiquitous computing. Oppnets bridge heterogeneous devices, networks, or systems under one umbrella, so that all their resources, such as computation, communication, sensing, actuation, storage, etc. can be integrated crossing boundaries imposed by technological discrepancies, such as programming language, hardware linguistics and communication medium. The distinguishing characteristic of Oppnets is the growth mechanism that is the incorporation of the resources of the devices, networks or systems. This way a task that could not be performed by a single device earlier, can now be accomplished by the Oppnet by delegating the task amongst the nodes with the resource capabilities to process the task (or sub-tasks).

This research entailed the design and implementation of a small-scale Oppnet named MicroOppnet, which not only serves as a proof of concept but is currently being extended as a testbed for designing, testing and implementing: (a) Oppnet primitives (cf. Appendix A); (b) routing, privacy and security protocols; and (c) Oppnet and ad hoc architectures.

The next research goal was the formulation of a Service Location and Planning (SLP) problem that can optimize use of services not only in Oppnets, but also in broader domains, such as traditional Service Oriented Computing (SOC) environments, and the more recent Cisco's AONs.

The novel SLP problem not only satisfies consumers' requests but also meets QoS guarantees of throughput and delay, and does it within the limits of underlying network link layer bandwidth capacities, and–most important to the providers– minimizing the service installation costs.

We have shown that the adapted Lagrangean Relaxation technique of the Service Location-Planning model yields a reduction of 17% on average in the number of constraints in the ILP model of the SLP problem.

7.2 Contributions Overview

The contributions of this research can be summarized as follows:

- Design and implementation of a small-scale Oppnet, called MicroOppnet
- Developing a proof-of-concept for Oppnets, which shows the feasibility of Oppnet
- Refining the idea of Oppnets
- Definition of a novel SLP problem
- SLP problem incorporates QoS constraints of throughput and delay

- SLP model is realistic as it incorporates underlying network link layer constraints, traffic splitting, service installation and local service installations
- Mathematical formulation of SLP as an Integer Linear Programming problem for small-scale networks
- Lagrangean Relaxation of the ILP formulation of the SLP problem for larger-scale networks, with the reductions of the number of constraints in ILP due to Lagrangean Relaxation by 17.42% on average.

7.3 Future Work

Future work on MicroOppnet includes: (a) extending the Class 2 opportunism communication by incorporating other communication media (cf. Table 4, Section 4.2) and the growth mechanism that's intrinsic to Oppnets; (b) designing privacy and security primitives and protocols; (c) scrutinizing and implementing opportunistic routing protocols of [63, 64]; (d) developing a Rapid Application Development (RAD) environment [47] for Oppnets.

Future work for SLP problem includes: (a) accounting for other QoS parameters such as reliability, security, adaptability, efficiency, etc; (b) developing a non-linear model for the SLP and comparing results with the ILP model; (c) further reducing the number of constraints by re-modeling the ILP formulation of the SLP problem in terms of binary variables; (d) comparing results from different Lagrangean relaxation techniques in terms of accuracy and complexity in implementation and execution time; (e) developing a simulation model for experimentation and validation of constraints and solutions of the ILP model for SLP; and (f) using evolutionary or genetic programming techniques as a meta-heuristic or heuristic for approximating SLP model for large-scale networks and comparing results with Lagrangean Relaxation approximation.

REFERENCES

[1] U. Hansmann, L. Merk, M. S. Nicklous and T. Stober, *Pervasive Computing, 2nd Edition.* Springer-Verlag Berlin Heidelberg, New York, NY, 2003.

[2] L. Pelusi, A. Passarella, and M. Conti, "Opportunistic Networking: Data Forwarding in Disconnected Mobile Ad Hoc Networks," *IEEE Communications*, Vol. 44(11), Nov. 2006, pp. 134-141.

[3] L. Lilien, Z. H. Kamal, V. Bhuse, and A. Gupta, "The Concept of Opportunistic Networks and Their Research Challenges in Privacy and Security," book chapter in: *Mobile and Wireless Network Security and Privacy* ed. by K. Makki et al., Springer Science+Business Media, Norwell, Massachusetts, 2007.

[4] L. Lilien, Z. H. Kamal, V. Bhuse, and A. Gupta, "Opportunistic Networks: The Concept and Research Challenges in Privacy and Security," *Proc. Intl. Workshop on Research Challenges in Security and Privacy for Mobile and Wireless Networks (WSPWN 2006)*, Miami, Florida, March 2006.

[5] B. Bhargava, L. Lilien, A. Rosenthal and M. Winslett, "PervasiveTrust," *IEEE Intelligent Systems*, vol. 19(5), Sep./Oct.2004, pp. 74-77.

[6] L. Lilien, A. Gupta, and Z. Yang "Opportunistic Networks for Emergency Applications and Their Standard Implementation Framework," *Proc. The First Intl. Workshop on Next Generation Networks for First Responders and Critical Infrastructure (NetCri07)*, New Orleans, Louisiana, April 2007.

[7] OnStar Corp., "On Star Explained," 2007. Last accessed on October 5, 2007.

Online: http://www.onstar.com/us_english/jsp/explore/index.jsp

[8] Cisco Systems Inc., "Introducing Cisco Application-Oriented Networking—A CIO Brief," 2006. Last accessed on October 5, 2007. http://www.cisco.com

[9] M. Baker, R. Buyya, and D. Laforenza, "Grids and Grid technologies for wide-area distributed computing," *Software—Practice & Experience*, Vol. 32(15), Dec. '02, pp. 1437–1466.

[10] IBM, "IBM Solutions Grid for Business Partners—Helping IBM Business Partners to Grid-enable applications for the next phase of e-business on demand," International Business Machines Corporation 2002, Austin, TX, 2002.

[11] Wikipedia contributors, "Grid computing," Wikipedia, The Free Encyclopedia, 28 June 2007. Last accessed October 5, 2007. http://en.wikipedia.org/w/index.php?title=Grid_computing&oldid=141246324

[12] I.F. Akyildiz, W. Su, Y. Sankarasubramaniam, and E. Cayirci, "Wireless sensor networks: a survey," *Computer Networks*, Vol. 38, 2002, pp. 393–422.

[13] P. Papadimitratos and Z. J. Haas, "Securing Mobile Ad Hoc Networks," In: Ilyas, M. (Hrsg.), *Handbook of Ad Hoc Wireless Networks*, CRC Press. 2002

[14] I.F. Akyildiz, X. Wang, and W. Wang, "Wireless mesh networks: A survey," *Computer Networks*, Vol. 47(4), March 2005, pp. 445-487.

[15] H. Karl, and A. Willig, "A short survey of wireless sensor networks," *Technical Report TKN-03-018*, Technical University Berlin, Berlin, Oct. 2003.

[16] L. Lilien, "Developing Specialized Ad Hoc Networks: The Case of Opportunistic Networks," *Proc. Workshop on Distributed Systems and Networks at the WWIC 2006 Conference*, Bern, Switzerland, May 2006.

[17] L. Lilien, "A Taxonomy of Specialized Ad Hoc Networks and Systems for Emergency Applications," *The First Intl. Workshop on Mobile and Ubiquitous Context Aware Systems and Applications (MUBICA 2007)*, Philadelphia, Pennsylvania, August 2007.

[18] R. Bruno, M. Conti, and E. Gregori, "Mesh Networks: Commodity Multi-hop Ad Hoc Networks," *IEEE Communications*, Vol. 43(3), March 2005, pp. 123-131.

[19] Wikipedia contributors, "Mesh networking," Wikipedia, The Free Encyclopedia, 21 June 2007. Last accessed online October 5, 2007. http://en.wikipedia.org/w/index.php?title=Mesh_networking&oldid=139745759

[20] B. Ahlgren, L. Eggert, B. Ohlman, and A. Schieder, "Ambient Networks: Bridging Heterogeneous Network Domains," *The 16th Annual IEEE International Symposium on Personal Indoor and Mobile Radio Communications (PIMRC)*, Berlin, Germany, 11-14 Sep. 2005.

[21] Ambient Networks, Last accessed on June 30, 2007. Online http://www.ambient-networks.org/

[22] N. Niebert, A. Schieder, H. Abramowicz, G. Malmgren, J. Sachs, U. Horn, C. Prehofer, H. Karl, "Ambient Networks – An Architecture for Communication Networks Beyond 3G," *IEEE Wireless Communications (Special Issue on 4G*

Mobile Communications – Towards Open Wireless Architecture), Vol. 11(2), April 2004, pp. 14-23.

[23] V. Cerf, S. Burleigh, A. Hooke, L. Torgerson, R. Durst, K. Scott, K. Fall, H. Weiss, "Delay-Tolerant Network Architecture," *DTN Research Group Internet Draft*, March 2003.

[24] L.M. Feeney, B. Ahlgren, and A. Westerlund, "Spontaneous Networking: An Application-oriented Approach to Ad Hoc Networking," *IEEE Communications Magazine*, Vol. 39(6), June 2001, pp. 176-181.

[25] Z. H. Kamal, L. Lilien, A. Gupta, Z. Yang, and M. Batsa, "Proof of Concept for Class 2 Opportunistic Networks – a New Paradigm for Unlicensed Mobile Access Technology," book chapter in: *Unlimited Mobile Access Technology: Protocols, Architectures, Security, Standards and Applications* ed. by Y. Zhang, et al., to appear.

[26] D. B. Shmoys, C. Swamy, and R. Levi, "Facility Location with Service Installation Costs," *Proceedings of the fifteenth annual ACM-SIAM symposium on Discrete algorithms*, 2004.

[27] M. G. C. Resende and R. F. Werneck, "A Hybrid Multistart Heuristic for the Uncapacitated Facility Location Problem," *AT&T Labs Research Technical Report*, TD-5RELRR (September 15), NJ, 2003.

[28] B. Aoun, R. Boutaba, Y. Iraqi, and G. Keyward, "Gateway Placement Optimization in Wireless Mesh Networks with QoS Constraints," *IEEE*

Communication, Vol. 24(11), Nov. 2006, pp. 2127-2136.

[29] Y. Bejerano, "Efficient Integration of Multihop Wireless and Wired Networks with QoS constraints," *IEEE/ACM Transactions on Networking*, Vol. 12(6), Dec. 2004, pp. 1064-1078.

[30] P. A. Bonatti, and P. Festa, "On Optimal Service Selection," *Proceedings of ACM International Conference on World Wide Web (WWW'05)*, Chiba, Japan, May 10-14 2005, pp. 530-538.

[31] J. Cardinal, and M. Hoefer, "Selfish Service Installation in Networks," *In Proc. of Int. Conf. on Internet and Network Economics (WINE'06)*, Vol. 4286, Lecture Notes in Computer Science, Springer-Verlag, 2006, pp. 174-185.

[32] R. Rizzi, and M. Rospocher, "Covering partially directed graphs with directed paths," *Discrete Mathematics*, Vol. 306(13), July 06, Elsevier B.V., Amsterdam, 2006, pp. 1390-1404.

[33] F. Zhu, M. Mutka, and L. M. Ni, "Service Discovery in Pervasive Computing," *IEEE Pervasive Computing*, Vol. 4(4), Oct–Dec. 2005, pp. 81-90.

[34] Atinav, Online at http://www.atinav.com, 2004-2006. Last accessed on June 30, 2007.

[35] B. Hopkins and R. Anthony, *Bluetooth for Java,* Apress, 2003.

[36] Crossbow Technology Inc., 2007. Last accessed on October 5, 2007 http://www.xbow.com/

[37] TinyOS, UC Berkeley, 2004. Last accessed on June 30, 2007,

http://www.tinyos.net/

[38] UC Berkeley WEBS Project, *nesC: A Programming Language for Deeply Networked Systems*, Dec. 2004. Last accessed on June 30, 2007 http://nescc.sourceforge.net/

[39] M. Berkelaar, K. Eikland and P. Notebaert, "lp_solve 5.5.0.10," Online, last accessed: http://lpsolve.sourceforge.net/5.5/

[40] Z. H. Kamal, A. Al-Fuqaha, and A. Gupta, "A service location problem with QoS constraints," *Proceedings of 2007 International Conference on Wireless Communications and Mobile Computing (IWCMC'07)*, Hawaii, USA, 2007, pp. 641-646.

[41] Wikipedia contributors, "LP relaxation," Wikipedia, The Free Encyclopedia, 3 July 2007, last accessed July 10, 2007. http://en.wikipedia.org/w/index.php?title=LP_relaxation&oldid=142290098

[42] B. Hunsaker, "IE 3051: Computational Optimization – Notes on Decomposition," University of Pittsburgh, Pittsburgh, PA. Online, last accessed July 10, 2007, http://www.engr.pitt.edu/hunsaker/3051/decomposition.pdf

[43] X. Zhao, P.B. Luh, and J. Wang, "Surrogate Gradient Algorithm for Lagrangian Relaxation," *Journal of Optimization Theory and Applications*, 100 (3), March 1999, pp. 699 – 712.

[44] D. M. Gay and L. Kaufman, "Tradeoffs in Algorithms for Separable Nonlinear Least Squares," *Proceedings of the 13th World Congress on Computational and*

Applied Mathematics (IMACS '91), edited by R. Vichnevetsky and J. J. H. Miller, Criterion Press, Dublin, 1991, pp. 157–158.

[45] M. L. Fisher, "The Lagrangian Relaxation Method For Solving Integer Programming Problems," *Management Science*, 27(1), January 1981, pp. 1 – 18.

[46] Michael A. Trick, "An application oriented tutorial on relaxations," Carnegie Mellon University, Pittsburgh, February, 1996. Last accessed online: http://mat.gsia.cmu.edu/mstc/relax/relax.html

[47] RAPIDware: Component-Based Development of Adaptable and Dependable Middleware, Network Systems (SENS) Laboratory, Michigan State University. Last accessed on June 30 2007. Online: http://www.cse.msu.edu/~mckinley/rapidware/

[48] K. Zhu, available online, last accessed September 9, 2007, http://networks.cs.ucdavis.edu/~zhuk/toplogies.html

[49] Z. H. Kamal, A. Gupta, L. Lilien, and Z. Yang, "The MicroOppnet Tool for Collaborative Computing Experiments with Class 2 Opportunistic Networks," *The 3rd Intl. Conf. on Collaborative Computing: Networking, Applications and Worksharing (CollaborateCom 2007)*, White Plains, New York, November 2007.

[50] Z. H. Kamal, A. Al-Fuqaha, and A. Gupta, "A Service Location and Planning problem with QoS constraints for Small and Large-scale Networks," submitted to *Special Issue of Elsevier Computer Communications on Self-Organization and*

Self-Management in Communications –Taking Vital Steps Towards Realizing Autonomic Networking, under review.

[51] Z. H. Kamal, A. Al-Fuqaha, and A. Gupta, "Service Location Planning in Large Scale Networks with QoS constraints," submitted to *2008 IEEE International Conference on Communications (ICC 2008)*, under review.

[52] J. B. Orlin, Online lecture notes, last accessed July 9, 2007. http://web.mit.edu/jorlin/www/15.082/lectures/19_lagrangian_Relaxation_1.ppt

[53] L. Lilien, A. Gupta, Z. H. Kamal, and Z. Yang, "Opportunistic Networks and Their Emergency Applications," submitted to *Special Issue of Pervasive and Mobile Computing Journal on Homeland and Global Security*, under review.

[54] L. Kleinrock, *Queueing Systems, Volume 1: Theory*, Wiley-Interscience, 1975.

[55] L. Lilien, Z. H. Kamal and A. Gupta, "Opportunistic Networks: Research Challenges in Specializing the P2P Paradigm," *Proc. 3rd Int. W. on P2P Data Management, Security and Trust (PDMST'06)*, Kraków, Poland, Sep. 2006, pp. 722-726.

[56] F. A. Chudak and D. P. Williamson, "Improved Approximation Algorithms for Capacitated Facility Location Problems," *Proceedings of the 7th International Conference on Integer Programming and Combinatorial Optimization (IPCO'99)*, Graz, Austria, June 1999.

[57] S. Jin, and L. Wang, "Content and Service Replication Strategies in Muli-hop Wireless Mesh Networks," *International Symposium on Modeling, Analysis and*

Simulation of Wireless and Mobile Systems (MSWiM'05), Montreal, Quebec, Canada, 2005.

[58] X. Gao, R. Jain, Z. Ramzan, and U. Kozat, "Resource Optimization for Web Service Compostion," *IEEE International Conference on Services Computing (SCC'05)*, Orlando, Florida, 2005.

[59] D. B. Shmoys, E. Tardos, and K. Aardal, "Approximation algorithms for facility location problems," *ACM Symposium Proceedings on Theory of Computing*, pp. 265–274, 1997.

[60] K. P. Bennet , A. Demiriz , J. Shawe-Taylor, "A Column Generation Algorithm For Boosting," *Proceedings of the Seventeenth International Conference on Machine Learning*, pp.65-72, 2000.

[61] S. Nash and A. Sofer, *Linear and Nonlinear Programming*. New York, NY: McGraw-Hill, 1996.

[62] A. Demiriz, K. P. Bennett, J. Shawe-Taylor, "Linear Programming Boosting via Column Generation," *Machine Learning*, Vol. 46 (1-3), pp. 225–254, 2002.

[63] Y. Wang, S. Jain, M. Martonosi and K. Fall, "Erasure-Coding Based Routing for Opportunistic Networks," *ACM Conf. of the Special Interest Group on Data Communication (SIGCOMM'05)*, Philadelphia, PA, Aug. 2005.

[64] P. Sistla, O. Wolfson, and B. Xu, "Opportunistic Data Dissemination in Mobile Peer-to-Peer Networks," *9th Intl. Symp. on Advances in Spatial and Temporal Databases (SSTD'05)*, Angra dos Reis, Brazil, Aug. 2005.

[65] ref for smoke detector image in Figure 9:

http://www.anaheim.net/depts_servc/fire/com_svc/smoked2.gif

[66] ref for webcam image in Figure 9:

http://www.marktmedia.nl/osc/images/webcam1.jpg

[67] ref for laptop in Figure 9:

http://www.nilkanth.com/my-uploads/dv6114laptop2.jpg

Appendix A

OVM Primitive

The Oppnet Virtual Machine (OVM) [6] is a part of the Oppnet. The goal of OVM project is to propose a standard for implementation of Oppnets. The standard will facilitate implementations from different software vendors and will assure their interoperability.

OVM will allow developing and marketing standard library routines and APIs to be used for implementing all kinds of Oppnet-based applications. OVM will not only facilitate application development but will also assure interoperability among different Oppnet implementations and third-party Oppnet products.

A list of goals for OVM includes the following:

- Design an application programming interface.
    - Provide extensions that allow greater flexibility.
    - Can be implemented on many vendor platforms.
    - Can be used in a heterogeneous environment.
- Allow efficient communication.
    - With uniformed data format.
    - Assume a reliable communication interface: the user need not cope with communication failures. Such failures are dealt with by the underlying communication subsystem.

This appendix details a list of primitives, divided into categories. It is summarized in Tables A1–A4. The procedures for Oppnet control center (CC), seeds, helpers, and lites have prefixes "CTRL_", "SEED_", "HLPR_", and "LITE_", respectively.

Table A1

Partial list of OVM primitives for CC nodes

Name of the Primitive	Functions of the Primitive
CTRL_initiate	Initiate Oppnet
CTRL_terminate	Terminate Oppnet
CTRL_command	Send commend to seed nodes

Table A2

Partial List of OVM Primitives for Seed Nodes

Name of the Primitive	Functions of the Primitive
SEED_scan	Scan communication spectrum to detect devices that could become candidate helpers
SEED_discover	Discover candidate helpers with a specific communication mechanism
SEED_listen	Receive and save messages in buffer
SEED_validate	Verify the received command
SEED_isMember	Checks if a device is already an Oppnet node (Oppnet member)
SEED_evaluateAdmit	Evaluate a device and admit it into Oppnet if the device meets criteria for admittance
SEED_sendTask	Send a task to other Oppnet device
SEED_delegateTask	Delegate a task that requires a permission from the delegating entity
SEED_release	Release a helper when no longer needed
SEED_processMsg	Process a message from buffer
SEED_report	Report information to control center/coordinator
SEED_update	Update a device in the Oppnet with new expectations
SEED_receiveTask	Receive task from control center or another seed
SEED_wait SEED_barrier	Wait for a certain amount of time become take another action Block the caller until all devices specified in the input parameter have called it

Table A3

Partial List of OVM Primitives for Helpers

Name of the Primitive	Functions of the Primitive
HLPR_isMember	Test if a helper is already a member of Oppnet
HLPR_joinOppnet	Join Oppnet
HLPR_scan	Scan communication spectrum to detect devices that could become candidate helpers (regular or lites)
HLPR_discover	Discover candidate helpers with a specified communication mechanism
HLPR_validate	Verify the received command
HLPR_switchMode	Switch between helpers' regular application and Oppnet application
HLPR_report	Send information/data to specified device
HLPR_selectTask	Select a task from the task queue to execute
HLPR_listen	Receive message and save it
HLPR_evaluateAdmit	Evaluate a candidate helper and admit it into Oppnet if it meets criteria defined by Oppnet
HLPR_runApplication	Execute application indicated by authorized Oppnet seed or helper node
HLPR_release	Release a helper (unless delegated a release task, a helper H can release only helpers admitted by H)
HLPR_processMsg	Process a message from buffer
HLPR_sendData	Send information/data to specified authorized Oppnet node
HLPR_leave	Inform a seed that the caller will quit Oppnet
HLPR_strongTask	Respond to the request sent from device and express the willingness to join Oppnet. By accepting this task, the device will abort previous task
HLPR_weakTask	Respond to the request sent from device and express the willingness to join Oppnet. By accepting this task, the device will put the task in a queue
HLPR_assignStrongTask	Assign tasks to a device. If accepted, the task will interrupt the previous task at the device
HLPR_assignWeakTask	Assign tasks to a device. If accepted, the task will be queued

Table A4

Partial List of OVM Primitives for Lites (Lightweight Helpers)

Name of the Primitive	Functions of the Primitive
LITE_isMember	Test if a lit is already a member of Oppnet
LITE_joinOppnet	Join Oppnet
LITE_validate	Verify the received command
LITE_switchMode	Switch between lites' regular application and Oppnet application
LITE_report	Send information/data to specified device
LITE_selectTask	Select a task from the task queue to execute
LITE_listen	Receive message and save it
LITE_runApplication	Execute application indicated by authorized Oppnet seed or helper node
LITE_processMsg	Process a message from buffer
LITE_sendData	Send information/data to specified authorized Oppnet node
LITE_leave	Inform a seed that the caller will quit Oppnet
LITE_strongTask	Respond to the request sent from device and express the willingness to join Oppnet. By accepting this task, the device will abort previous task
LITE_weakTask	Respond to the request sent from device and express the willingness to join Oppnet. By accepting this task, the device will put the task in a queue

Appendix B

MicroOppnet Code

The code presented in this section refers to the devices in Figure 4.

Oppnet Seed Code

Code on Laptop A :

1. OppComm.java contains classes: OppComm, ClientApp_Copy, ServerSideApp
2. humaBcastInject.java contains classes: humaBcastInject

Note: humaBcastInject refers to other classes that are used as-is with the TinyOS operating system found at http://www.tinyos.net/

OppComm.java

```
package net.tinyos.tools;

import javax.swing.*;
import java.io.*;
import java.awt.image.*;
import javax.imageio.*;
import com.atinav.standardedition.io.*;

// bluetooth related imports
import javax.bluetooth.*;
import javax.obex.*;

/**
 * Originally ised sample code from source below then modified by Zill-E-Huma Kamal
 * <p>Title: ServerApp.java</p>
 * <p>Description: Sample OBEX Object Push server application using JSR 82 API</p>
 * <p>Copyright: Copyright(c) Atinav Inc. 2002</p>
 * @version 1.0
 */

class ServerSideApp extends ServerRequestHandler {

        public String file_recvd = "";

    public ServerSideApp() {
    }

    public void server() throws IOException {
        try {
            LocalDevice ld = LocalDevice.getLocalDevice();
            System.out.println("BD_Address " + ld.getBluetoothAddress());
            System.out.println("Name " + ld.getFriendlyName());
```

```
            System.out.println("Discoverable = " + ld.getDiscoverable());

        }catch(Exception e) {
            System.out.println("Error getting local device"+e.getMessage());
        }

        try {    // btgoep://[localhost]:UUID[;][name=<service name>]

            SessionNotifier sn = (SessionNotifier)Connector.open("btgoep://localhost:1105;name=OBEX");
            ServiceRecord sr = LocalDevice.getLocalDevice().getRecord(sn);
            DataElement de = new DataElement(DataElement.DATSEQ);

            //Register Push Service attribute
            DataElement suppfeature = new DataElement(DataElement.U_INT_1, 0xFF);
            de.addElement(suppfeature);
            sr.setAttributeValue(0x0303, de);

            System.out.println("Waiting to process client requests...");
            sn.acceptAndOpen(this);
            System.out.println("out of accept and open");
        } catch (IOException ex) {
            System.out.println("ERROR: Failed opening connection");
            return;
        }

                    try
                    {
                    synchronized(this){
                                this.wait();
        }}catch(Exception e){System.out.println("Error: " + e.toString());}

        System.out.println("end of server method");

    }//end of method server()

    public int onConnect(HeaderSet request, HeaderSet reply) {
        System.out.println("\n\nOnConnect\n\n");
        //save the request packet details if neccessary
        createHeaderSet();
        System.out.println("\n\nend of OnConnect\n\n");
        return  ResponseCodes.OBEX_HTTP_OK;

    }//end of method onConnect

    public int onPut(Operation op) {
        System.out.println("\n\nIn OnPut method\n\n");

        boolean textFile = false;                    //false =:> image file to be read and written  true=:> text file to
be read and written

        try {
            InputStream in = op.openInputStream();
            HeaderSet hdr = op.getReceivedHeaders();

                                File f = new File((String)hdr.getHeader(HeaderSet.NAME));

            //System.out.println((String)hdr.getHeader(HeaderSet.NAME));
            System.out.println("file name = " + f.getName());

                                if(f.getName().toLowerCase().endsWith(".txt"))
                                        textFile = true;
                                else if
(f.getName().toLowerCase().endsWith(".jpg")||f.getName().toLowerCase().endsWith(".jpeg"))
                                        textFile = false;
                                else
                                {
                                        System.out.println("The file type to be received is not currently supported.
Exiting application.");
                                        System.exit(0);
                                }//file type currently not supported


                                if(textFile)
                                {

                                        int data = 0;
                                        System.out.print("**** Data received from client using OBEX Put ****\nReceived
data => ");

                                        while((data = in.read()) != -1)
                                        {
                                                System.out.print((char)data);
                                                file_recvd += (char)data;

                                        }//end of while reading all of received file
```

```
                            System.out.println("\n______________________________________________\n");

                            writeFile(f, file_recvd);

                    }//read text file
                    else
                    {
                            byte[] imgData = null;

                    try{
                            int length = in.read() << 8;
                            length |= in.read();

                            if (length <= 0) {
                                    throw new IOException("Can't read a length");
                            }

                            // read the image now
                            imgData = new byte[length];
                            length = 0;

                            while (length != imgData.length) {
                                    int n = in.read(imgData, length, imgData.length - length);

                                    if (n == -1) {
                                            throw new IOException("Can't read a image data");
                                    }
                                    length += n;
                            }
                            //in.close();
                    } catch (IOException e) {
                            System.err.println("Can't read from server for: " + e);
                    }

                    }//if image received

                    System.out.println("received everything....");
                    in.close();
                    op.close();

    } catch(IOException ioe){
        System.out.println("ERROR: IOException in onPut in app");
        return ResponseCodes.OBEX_HTTP_INTERNAL_ERROR;
    }

            System.out.println("\n\nend of onPut\n\n");


            synchronized(this){
                    this.notify();
        }

    return ResponseCodes.OBEX_HTTP_OK;
}//end of method onPut

public static boolean writeFile (File file, String dataString) {

                try {
                        PrintWriter out = new PrintWriter (new BufferedWriter (new FileWriter (file)));
                        out.print (dataString);
                        System.out.println("string received written to file");
                        out.flush ();
                        out.close ();
                }
                catch (IOException e) {
                        return false;
                }

                return true;
      }// end of method writeFile

      public static boolean writeImage (File file, ByteArrayInputStream rawImageBytes) {

              try{
                        BufferedImage image = ImageIO.read (rawImageBytes);

                        ImageIO.write( image, "jpg", file);

               }catch(IOException ioe){System.out.println("Error reading image: "+ioe.toString());}

                return true;
      }// end of method writeImage

      private byte[] getImageData(String imgName) {
```

```
                if (imgName == null) {
                        return null;
                }
                InputStream in = getClass().getResourceAsStream(imgName);

                // read image data and create a byte array
                byte[] buff = new byte[1024];
                ByteArrayOutputStream baos = new ByteArrayOutputStream(1024);

                try {
                        while (true) {
                                int length = in.read(buff);

                                if (length == -1) {
                                        break;
                                }
                                baos.write(buff, 0, length);
                        }
                } catch (IOException e) {
                        System.err.println("Can't get image data: imgName=" + imgName + " :"
                                        + e);
                        return null;
                }
                return baos.toByteArray();

        }//end of method getImageData


}//end of class ServerSideApp

/***************************************************** END OF ServerSideApp FILE************************************/



/*************************************************** CLIENT APP COPY FILE
***********************************************/

/**
 * Originally ised sample code from source below then modified by Zill-E-Huma Kamal
 * <p>Title: ClientApp.java</p>
 * <p>Description: Sample OBEX Object Push client application using JSR 82 API</p>
 * <p>Copyright: Copyright(c) Atinav Inc. 2002</p>
 * @version 1.0
 */

class ClientApp_Copy extends ServerRequestHandler implements DiscoveryListener {
    private boolean inquiryCompleted    = false;
    private boolean authenticate        = false;
    private RemoteDevice[] devices =  new RemoteDevice[15];
    private static int count = 0;
    private String connectionURL = null;
    private ServiceRecord[] records = null;
    private DiscoveryAgent da;

          public File fFile;

    public ClientApp_Copy() {

    }

    public String[][] getDeviceInfo(){

        System.out.println("What is count? count = " + count);
        String[][] pairedFriendlyAndAddress = new String[count+1][count];
        System.out.println("What is count? count = " + count);
        System.out.println("*****Another attempt to get friendly name");

        for(int i = 0; i < count; i++){

            try {

                pairedFriendlyAndAddress[0][i] = devices[i].getFriendlyName(true);
                pairedFriendlyAndAddress[1][i] = devices[i].getBluetoothAddress();

            }catch(IOException e){System.out.println(e.getMessage());}

        }//end of for loop

        System.out.println("Printing list of paired...");

        for(int j = 0; j < count; j++)
            System.out.println(pairedFriendlyAndAddress[0][j] + "\t" + pairedFriendlyAndAddress[1][j]);
```

```
        return pairedFriendlyAndAddress;

    }//end of method getDeviceInfo

    public void client(String url) throws IOException {
        HeaderSet hdr = null;
                boolean imageFlag = false;          //false =:> text file to send          true =:> image to
be sent

                if(fFile.exists())
                        System.out.println("file EXISTS: "+fFile.getName()+" at "+fFile.getPath());
                else
                {
                        System.out.println("The file you selected doesnot exist. Quitting application.");
                        System.exit(0);
                }

                String filename = fFile.getName();

        System.out.println("Connection URL -> " + url);
        if (url == null) {
            System.out.println("ERROR: Null URL received, quitting app...");
            System.exit(0);
        }
        ClientSession cs = null;
        try {
            System.out.println("url = " + url + " => this is the BT address of the machine I am connecting to!!!\n");
            cs = (ClientSession)Connector.open(url);
            hdr = cs.createHeaderSet();

            System.out.println("Invoking obex connect...");

            hdr = cs.connect(hdr);      //connect first thing client does after server registers its services
internally

            if (hdr.getResponseCode() != ResponseCodes.OBEX_HTTP_OK) {
                System.out.println("ERROR: Connection request failed");
                return;
            }
        } catch (IOException ex) {
            System.out.println("ERROR: IOException caught --- " +ex.toString());
            return;
        }catch (Exception x) {
            System.out.println("ERROR: Exception caught");
            x.printStackTrace();
            return;
        }

                System.out.println("getting ready to read file: "+filename);

                hdr = cs.createHeaderSet();
            hdr.setHeader(HeaderSet.NAME, filename);
                //adding .txt extension made it possible for Nokia 6103 to view/open file at Nokia end
                //but still doesnot work for Motorola

                String ext = filename.toLowerCase();

        if(ext.endsWith(".jpg"))
        {
                        hdr.setHeader(HeaderSet.TYPE, "image/jpeg");
                        imageFlag = true;
                }
                else if(ext.endsWith(".txt"))
                {
                        hdr.setHeader(HeaderSet.TYPE, "text/plain");
                        imageFlag = false;
                }
                else
                {
                        System.out.println("Currently the file format that you want to send is not supported.
Application closing...");
                        return;
                }

                if(imageFlag)
                {
                        byte[] buffer = null;
                        try {
                                BufferedImage image = ImageIO.read(fFile);

                                ByteArrayOutputStream bos = new ByteArrayOutputStream();
                                ImageIO.write(image, "jpg", bos);
                                buffer = bos.toByteArray();
                        } catch (Exception e) {
                                        e.printStackTrace();
```

```
            }
            hdr.setHeader(HeaderSet.LENGTH, new Long(buffer.length));

            Operation po = cs.put(hdr);
            OutputStream os = po.openOutputStream();

            int of=0;
            do{
                    int le = 64;
                    if(le+of>buffer.length)
                            le=buffer.length-of;

                    os.write(buffer,of,le );
                    os.flush();
                    of+=le;
            }while(of<buffer.length);

            po.close();

        }//if image file

        else
        {
            String msg = readFile(fFile);
            Operation op = null;
            try{
                    op = cs.put(hdr);
                    System.out.println("AFTER PUT in obex APP");
            }catch(Exception ioe){
                    System.out.println("Failed putting data to the server");
                    ioe.printStackTrace();
                    cs.close();
                    return;
            }

            OutputStream out = null;
            out = op.openOutputStream();
            try{

                    out.write(msg.getBytes());

            }catch(IOException ioe){

                    System.out.println("Failed putting data to the server");
                    op.close();
                    cs.close();
                    return;
            }

            System.out.println("Data successfully pushed to the server");

            // closing the streams
            out.close();
            op.close();

        }//otherwise send text

        cs.disconnect(null);

        // closing the session
        cs.close();

}//end of method client

    public String readFile(File file) {

        StringBuffer fileBuffer;
        String fileString=null;
        String line;

        try {
            FileReader in = new FileReader(file);
            BufferedReader dis = new BufferedReader(in);
            fileBuffer = new StringBuffer() ;

            while ((line = dis.readLine()) != null) {
                fileBuffer.append(line + "\n");
            }

            in.close();
            fileString = fileBuffer.toString();
        } catch (IOException e ) {
            return null;
        }
```

```
            return fileString;
 } // end of method readFile

 /*
 * method client_image extracted from http://www.mobile-j.de/snipsnap/space/Fun/Send+PC+screen+to+SU2
 * edited by Zill-E-Huma Kamal to fit application purpose
 * removed client_image instead used part of it in method client
 */

public int onGet(Operation op) {

    try{
        //The server has received a GET request for client.
        System.out.println("Received a GET request from client.... . ");
        HeaderSet hdr = op.getReceivedHeaders();

        System.out.println("Server has received a request for the file "+
                (hdr.getHeader(HeaderSet.NAME)).toString());
        String url = "file://name=" +
                (hdr.getHeader(HeaderSet.NAME)).toString() + ";mode=r";
        InputConnection inpcon =
                (InputConnection)Connector.open(url);
        InputStream in = inpcon.openInputStream();
        byte[] fileAsBytes     = new byte[97];
        in.read(fileAsBytes);
        System.out.println("File read fully into the port.... ");
        for(int i =0; i<fileAsBytes.length; i++)
            System.out.print((char)fileAsBytes[i]);
        DataOutputStream out = op.openDataOutputStream();
        out.write(fileAsBytes, 0, fileAsBytes.length);
        System.out.println("\n" + "File written back to client.... ");
        op.close();
        in.close();
    } catch(IOException e){
        System.out.println(e.getMessage());
    } catch(ArrayIndexOutOfBoundsException e){
        System.out.println(e.getMessage());
    }

    return ResponseCodes.OBEX_HTTP_OK;

}//end of onGet method

public int onConnect(HeaderSet request, HeaderSet reply) {
    //save the request packet details if neccessary
    createHeaderSet();
    return  ResponseCodes.OBEX_HTTP_OK;
}//end of method onConnect

public void searchDevices() throws Exception {
    LocalDevice ld = LocalDevice.getLocalDevice();
    da = ld.getDiscoveryAgent();

    System.out.println("Starting device inquiry...");
    da.startInquiry(DiscoveryAgent.GIAC, this);

    // wait till the device enquiry is completed
    synchronized(this){
        this.wait();
    }
}//end of method searchDevices

public String connectToBTdevice(String bt_device) throws Exception {
    System.out.println("you are connecting to " + bt_device + " from method connectToBTdevice\n");

    int[] attrSet = {0,3,4};
    UUID[] uuids = new UUID[1];
    uuids[0] = new UUID("1105", true);
    int i, index;

    System.out.println("\nSearching for Service...\n");

    for(i = 0; i < count; i++) {
        if(devices[i].getBluetoothAddress().equals(bt_device)) {
            index = i;
            break;
        }
    }//end of for loop
    try{
        System.out.println("\nSearching for Service @ " + devices[i].getBluetoothAddress());
        int transactionid = da.searchServices(attrSet, uuids, devices[i], this);
        if(transactionid != -1) {
            synchronized(this) {
                this.wait();
            }
```

```
            }
            if(connectionURL != null){
                return connectionURL;
            }
        }catch(Exception e) {
            System.out.println(e.getMessage());
        }
        return null;
    }//end of method conncetToBTDevice

    public synchronized void inquiryCompleted(int discType) {
        this.notify();
    }

    public synchronized void deviceDiscovered(RemoteDevice btDevice, DeviceClass cod) {

        //trying not to add same device to RemoteDevice list: devices

        boolean flag_found = false;

        if(count > 0) {

            for(int i =0; i < count; i++) {
                flag_found = false;

                System.out.println("*** compare: " + devices[i].getBluetoothAddress() + " with " +
btDevice.getBluetoothAddress() + "\n");


                if(devices[i].getBluetoothAddress().equals(btDevice.getBluetoothAddress())) {

                    System.out.println("**>>> same bt device found <<**\n");

                    flag_found = true;

                    break;

                }//end of if

            }//end of for
        }//if more than one device


        //if not already in list add it in list
        if(flag_found == false) {

            devices[count++] = btDevice;
            System.out.println("New Device discovered : "+ btDevice.getBluetoothAddress());

        }//if not already found add to list

    }//end of method deviceDiscovered

    public synchronized void servicesDiscovered(int transID, ServiceRecord[] servRecords) {
        if(servRecords.length == 0) {
            synchronized(this){
                this.notify();
            }

        }
        records = new ServiceRecord[servRecords.length];
        records = servRecords;
        for(int i=0;i<servRecords.length;i++){
            connectionURL = servRecords[i].getConnectionURL(1,true);
            System.out.println("Connection url :"+connectionURL);
            if(connectionURL != null){
                synchronized(this){
                    this.notify();
                }
                break;
            }
        }
    }

    public synchronized void serviceSearchCompleted(int transID, int respCode) {
        if(respCode==SERVICE_SEARCH_ERROR){
            System.out.println("\nSERVICE_SEARCH_ERROR\n");
        }
        if(respCode==SERVICE_SEARCH_COMPLETED){
            System.out.println("\nSERVICE_SEARCH_COMPLETED\n");
        }
        if(respCode==SERVICE_SEARCH_TERMINATED){
            System.out.println("\n SERVICE_SEARCH_TERMINATED\n");
        }
        if(respCode == SERVICE_SEARCH_NO_RECORDS){
```

```
            System.out.println("\n SERVICE_SEARCH_NO_RECORDS\n");
        }
        if(respCode == SERVICE_SEARCH_DEVICE_NOT_REACHABLE){
            System.out.println("\n SERVICE_SEARCH_DEVICE_NOT_REACHABLE\n");
        }

        synchronized(this){
            this.notify();
        }
    }//end of method serviceSearchCompleted

}//end of class ClientApp_Copy

/*********************************** END OF CLIENT APP COPY FILE  ****************************************/



/*
 * BTGuiFrame_OBEX.java   << frame created in netbeans
 * Created on February 6, 2006, 1:32 PM
 */


class OppComm {

    private humaBcastInject bi = new humaBcastInject();
    private ServerSideApp s = new ServerSideApp();
    private ClientApp_Copy m = new ClientApp_Copy();
    private String url="";
    private String[][] output;
    private String bt_device_address ="";
    private boolean smode = true;       //false =:> client and true =:> server

    public String file_recvd_string="";
    public boolean fwd_msg = false;
    public boolean bt = false;


    public OppComm() {

    }//end of constructor


    public void serverActionPerformed(java.awt.event.ActionEvent evt) {

        System.out.println("acting as server... getting ready to receive files... ");

        smode = true;       //false =:> client and true =:> server

                    bt = true;

        if(smode) {
                       try {

                              s.server();

                              System.out.println("\n\n\n\nout of server and message received is:\n"+s.file_recvd);

                              System.out.println("############## getting ready to push distress signal to sensornet
##############");

                              String[] cmd = {"forward_msg", s.file_recvd};

                                        fwd_msg = true;
                              bi.startInjection(cmd);

                        } catch (IOException ex) {
                                        System.out.println("ERROR: IOException caught  " + ex.getMessage());
                               return;
                    }

          }//end of if server mode then call server method

        smode = false;

    }//end of method serverActionPerformed


    public boolean getFwdMsg(){
          return fwd_msg;
    }
```

```
    public boolean getBT(){
          return bt;
    }

    private void clientActionPerformed(java.awt.event.ActionEvent evt) {

        System.out.println("acting as client... getting ready to send files... ");

        smode = false;

    }//end of clientActionPerformed

    private void sendActionPerformed(java.awt.event.ActionEvent evt) {

        if(!smode){
           try {
              m.client(url);
             } catch (IOException ex) {
                 System.out.println("ERROR: IOException caught  " + ex.getMessage());
                 return;
             }

            smode = false;
        }//end of if in client mode
    }//end of sendActionPerformed

    public void ExitItemActionPerformed(java.awt.event.ActionEvent evt) {
           System.exit(0);
    }//end of ExitItemActionPerformed

}//end of class OppComm
```

humaBcastInject.java

```java
/* Modified code provided by Source below. Modified by Zill-E-Huma Kamal*/

// $Id: BcastInject.java,v 1.6.2.4 2003/08/21 01:15:18 cssharp Exp $

/*
 * "Copyright (c) 2000-2003 The Regents of the University  of California.
 * All rights reserved.
 *
 * Permission to use, copy, modify, and distribute this software and its
 * documentation for any purpose, without fee, and without written agreement is
 * hereby granted, provided that the above copyright notice, the following
 * two paragraphs and the author appear in all copies of this software.
 *
 * IN NO EVENT SHALL THE UNIVERSITY OF CALIFORNIA BE LIABLE TO ANY PARTY FOR
 * DIRECT, INDIRECT, SPECIAL, INCIDENTAL, OR CONSEQUENTIAL DAMAGES ARISING OUT
 * OF THE USE OF THIS SOFTWARE AND ITS DOCUMENTATION, EVEN IF THE UNIVERSITY OF
 * CALIFORNIA HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
 *
 * THE UNIVERSITY OF CALIFORNIA SPECIFICALLY DISCLAIMS ANY WARRANTIES,
 * INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY
 * AND FITNESS FOR A PARTICULAR PURPOSE.  THE SOFTWARE PROVIDED HEREUNDER IS
 * ON AN "AS IS" BASIS, AND THE UNIVERSITY OF CALIFORNIA HAS NO OBLIGATION TO
 * PROVIDE MAINTENANCE, SUPPORT, UPDATES, ENHANCEMENTS, OR MODIFICATIONS."
 *
 * Copyright (c) 2002-2003 Intel Corporation
 * All rights reserved.
 *
 * This file is distributed under the terms in the attached INTEL-LICENSE
 * file. If you do not find these files, copies can be found by writing to
 * Intel Research Berkeley, 2150 Shattuck Avenue, Suite 1300, Berkeley, CA,
 * 94704.  Attention:  Intel License Inquiry.
 */


/**
 *
 *
 * @author <a href="mailto:szewczyk@sourceforge.net">Robert Szewczyk</a>
 */
package net.tinyos.tools;

import net.tinyos.util.*;
import java.io.*;
import java.util.Properties;
import net.tinyos.message.*;

public class humaBcastInject implements MessageListener {
    static Properties p = new Properties();
    public static final byte LED_ON = 1;
    public static final byte LED_OFF = 2;
    public static final byte RADIO_LOUDER = 3;
    public static final byte RADIO_QUIETER = 4;
    public static final byte FORWARD_MSG = 7;
    public static final byte RETRIEVE_MSG = 8;

    public boolean read_log_done = false;

    //added
    public static String message = "";
    public boolean ret_msg_done = false;

    public static final short TOS_BCAST_ADDR = (short) 0xffff;

    public static void usage() {
          System.err.println("Usage: java net.tinyos.tools.BcastInject"+
                             " <command> [arguments]");
          System.err.println("\twhere <command> and [arguments] can be one of the following:");
          System.err.println("\t\tled_on");
          System.err.println("\t\tled_off");
          System.err.println("\t\tradio_louder");
          System.err.println("\t\tradio_quieter");
          System.err.println("\t\tforward_msg [message]");
          System.err.println("\t\tretrieve_msg ");
    }

    public static byte restoreSequenceNo() {
          try {
```

```
            FileInputStream fis = new FileInputStream("bcast.properties");
            p.load(fis);
            byte i = (byte)Integer.parseInt(p.getProperty("sequenceNo", "1"));
            fis.close();
            return i;
        } catch (IOException e) {
            p.setProperty("sequenceNo", "1");
            return 1;
        }
}
public static void saveSequenceNo(int i) {
        try {
            FileOutputStream fos = new FileOutputStream("bcast.properties");
            p.setProperty("sequenceNo", Integer.toString(i));
            p.store(fos, "#Properties for BcastInject\n");
        } catch (IOException e) {
            System.err.println("Exception while saving sequence number" +
                               e);
            e.printStackTrace();
        }
}

public static void setMessage(String msg)
{
                message = msg;
        }

        public static void startInjection(String[] argv) throws IOException{

        String cmd;
        byte sequenceNo = 0;
        boolean read_log = false;
        boolean ret_msg = false;

        if (argv.length < 1) {
            usage();
            System.exit(-1);
        }

        cmd = argv[0];

        SimpleCmdMsg packet = new SimpleCmdMsg();

        sequenceNo = restoreSequenceNo();
        packet.set_seqno(sequenceNo);
        packet.set_hop_count((short)0);
        packet.set_source(0);

        if (cmd.equals("led_on")) {
            packet.set_action(LED_ON);
        } else if (cmd.equals("led_off")) {
            packet.set_action(LED_OFF);
        } else if (cmd.equals("radio_louder")) {
            packet.set_action(RADIO_LOUDER);
        } else if (cmd.equals("radio_quieter")) {
            packet.set_action(RADIO_QUIETER);
        }
        //added
        else if (cmd.equals("retrieve_msg")) {
                  packet.set_action(RETRIEVE_MSG);
                  ret_msg = true;
        } else if (cmd.equals("forward_msg")) {
                  System.err.println("forward message");
                  packet.set_action(FORWARD_MSG);
                  setMessage(argv[1]);
                  packet.setString_args_untyped_args(message);
                  System.err.println("set the untyped args. hope this works, with message: " + message);
                  System.err.println("\n\nMessage going out is: \n"+packet);
                  System.err.println("\n\nMessage going out is: \n"+packet.getString_args_untyped_args());
        } else {
            usage();
            System.exit(-1);
        }

        try {
            System.err.print("Sending payload: ");

            for (int i = 0; i < packet.dataLength(); i++) {
                        System.err.print(Integer.toHexString(packet.dataGet()[i] & 0xff)+ " ");
            }
            System.err.println();

            MoteIF mote = new MoteIF(PrintStreamMessenger.err);

            // Need to wait for a retrieve_msg message to come back
```

```
                    humaBcastInject oc = null;
                    if (ret_msg) {
                              oc = new humaBcastInject ();
                              mote.registerListener(new SimpleCmdMsg(), oc);
                    }

                    mote.send(TOS_BCAST_ADDR, packet);

                    if (ret_msg) {
                              synchronized (oc) {
                                        if (oc.ret_msg_done == false) {
                                                  System.err.println("Waiting for response to ret_msg...");
                                                  oc.wait(10000);
                                        }
                                        if (oc.ret_msg_done == false) {
                                                  System.err.println("Warning: Timed out waiting for response to
ret_msg command!");
                                        }
                              }
                    }

              saveSequenceNo(sequenceNo+1);

          } catch(Exception e) {
              e.printStackTrace();
          }

    }

     public void messageReceived(int dest_addr, Message m) {

                              SimpleCmdMsg lm = (SimpleCmdMsg) m;

                              System.err.println("Received a message: "+lm+"\n\nimp area:
"+lm.getString_args_untyped_args());
                              synchronized (this) {
                                        ret_msg_done = true;
                                        this.notifyAll();
                              }
          }//end of messageReceived

}//end of humaBcastInject
```

Code on Sensornet base station BS_1 is the TinyOS application TOSBase and can be found at http://www.tinyos.net/tinyos-1.x/apps/TOSBase/

Oppnet helper code

Code on Wireless Sensor Network nodes contains two TinyOS applications SimpleCmd and Bcast, which have been modified by Zill-E-Huma Kamal to fit our purpose. This contains: Bcast.nc, BcastM.nc, ProcessCmd.nc, SimpleCmd.nc, SimpleCmdM.nc and SimpleCmdMsg.h

<u>Bcast.nc</u>

```
// $Id: Bcast.nc,v 1.2.14.4 2003/08/26 09:08:06 cssharp Exp $

/*
 * "Copyright (c) 2000-2003 The Regents of the University  of California.
 * All rights reserved.
 *
 * Permission to use, copy, modify, and distribute this software and its
 * documentation for any purpose, without fee, and without written agreement is
 * hereby granted, provided that the above copyright notice, the following
 * two paragraphs and the author appear in all copies of this software.
 *
 * IN NO EVENT SHALL THE UNIVERSITY OF CALIFORNIA BE LIABLE TO ANY PARTY FOR
 * DIRECT, INDIRECT, SPECIAL, INCIDENTAL, OR CONSEQUENTIAL DAMAGES ARISING OUT
 * OF THE USE OF THIS SOFTWARE AND ITS DOCUMENTATION, EVEN IF THE UNIVERSITY OF
 * CALIFORNIA HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
 *
 * THE UNIVERSITY OF CALIFORNIA SPECIFICALLY DISCLAIMS ANY WARRANTIES,
 * INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY
 * AND FITNESS FOR A PARTICULAR PURPOSE.  THE SOFTWARE PROVIDED HEREUNDER IS
 * ON AN "AS IS" BASIS, AND THE UNIVERSITY OF CALIFORNIA HAS NO OBLIGATION TO
 * PROVIDE MAINTENANCE, SUPPORT, UPDATES, ENHANCEMENTS, OR MODIFICATIONS."
 *
 * Copyright (c) 2002-2003 Intel Corporation
 * All rights reserved.
 *
 * This file is distributed under the terms in the attached INTEL-LICENSE
 * file. If you do not find these files, copies can be found by writing to
 * Intel Research Berkeley, 2150 Shattuck Avenue, Suite 1300, Berkeley, CA,
 * 94704.  Attention:  Intel License Inquiry.
 */
/*
 * Author: Alec Woo, David Culler, Robert Szewczyk, Su Ping
 *
 * $\Id$
 */

/**
 * @author Alec Woo
 * @author David Culler
 * @author Robert Szewczyk
 * @author Su Ping
 */

includes SimpleCmdMsg;

/**
 *
 * This configuration module wires BcastM module to components:
```

```
 * Main, SimpleCmd, GenericComm and LedsC.
 */

configuration Bcast { }
implementation {
  components Main, BcastM, SimpleCmd, GenericComm as Comm, LedsC;

  Main.StdControl -> BcastM;

  BcastM.Leds -> LedsC;

  BcastM.ProcessCmd-> SimpleCmd.ProcessCmd;
  BcastM.CommControl -> Comm;
  BcastM.SendCmdMsg -> Comm.SendMsg[AM_SIMPLECMDMSG];
  BcastM.ReceiveCmdMsg -> Comm.ReceiveMsg[AM_SIMPLECMDMSG];

}
```

BcastM.nc

```
//modified by Zill-E-Huma Kamal

// $Id: BcastM.nc,v 1.2.14.3 2003/08/26 09:08:06 cssharp Exp $
/*
 * "Copyright (c) 2000-2003 The Regents of the University  of California.
 * All rights reserved.
 *
 * Permission to use, copy, modify, and distribute this software and its
 * documentation for any purpose, without fee, and without written agreement is
 * hereby granted, provided that the above copyright notice, the following
 * two paragraphs and the author appear in all copies of this software.
 *
 * IN NO EVENT SHALL THE UNIVERSITY OF CALIFORNIA BE LIABLE TO ANY PARTY FOR
 * DIRECT, INDIRECT, SPECIAL, INCIDENTAL, OR CONSEQUENTIAL DAMAGES ARISING OUT
 * OF THE USE OF THIS SOFTWARE AND ITS DOCUMENTATION, EVEN IF THE UNIVERSITY OF
 * CALIFORNIA HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
 *
 * THE UNIVERSITY OF CALIFORNIA SPECIFICALLY DISCLAIMS ANY WARRANTIES,
 * INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY
 * AND FITNESS FOR A PARTICULAR PURPOSE.  THE SOFTWARE PROVIDED HEREUNDER IS
 * ON AN "AS IS" BASIS, AND THE UNIVERSITY OF CALIFORNIA HAS NO OBLIGATION TO
 * PROVIDE MAINTENANCE, SUPPORT, UPDATES, ENHANCEMENTS, OR MODIFICATIONS."
 *
 * Copyright (c) 2002-2003 Intel Corporation
 * All rights reserved.
 *
 * This file is distributed under the terms in the attached INTEL-LICENSE
 * file. If you do not find these files, copies can be found by writing to
 * Intel Research Berkeley, 2150 Shattuck Avenue, Suite 1300, Berkeley, CA,
 * 94704.  Attention:  Intel License Inquiry.
 */
/*
 * Author: Alec Woo, David Culler, Robert Szewczyk, Su Ping
 *
 * $\Id$
 */

/**
 * BcastM is a Tinyos application module.
 *
 * When a message with AM message type 8 is received from underlying
 * Comm layer, this module checks if it has seen the message before and
 * drops the message if so. Otherwise, it calls the local ProcessCmd
 * interface to execute the command in the message.
 * If the command is successful, it broadcasts the original message over
 * RF link.
 * @author Alec Woo
 * @author David Culler
 * @author Robert Szewczyk
 * @author Su Ping
 **/
includes SimpleCmdMsg;

module BcastM {
    provides        interface StdControl;
    uses {
      interface ProcessCmd;
      interface Leds;
      interface Pot;
      interface ReceiveMsg as ReceiveCmdMsg;
      interface SendMsg as SendCmdMsg;
      interface StdControl as CommControl;
    }
}

/* Module Implementation */

implementation
{
  TOS_MsgPtr msg;
  int8_t bcast_pending;
  TOS_Msg buf;
  int8_t lastSeqno;

  /**
   * Task:  Broadcast a message
   * @return Always returns <code>SUCCESS</code>
   **/

  task void forwarder() {
```

```
  call SendCmdMsg.send(TOS_BCAST_ADDR, sizeof(struct SimpleCmdMsg), msg);
}

/* Reset the bcast_pending flag to 0 if pmsg was sent successfully
 * @return Returns the value of 'status'
 **/
event result_t SendCmdMsg.sendDone(TOS_MsgPtr pmsg, result_t status) {
  if (status == SUCCESS) bcast_pending = 0;
  return status;
}

/**
 * Initalize the application.
 * @return A boolean indicating success or failure of the application
 * initialization.
 **/
command result_t StdControl.init() {
  msg = &buf;
  bcast_pending = 0;
  lastSeqno=0;

  return (call CommControl.init());
}
/** start generic communication interface **/
command result_t StdControl.start(){
  return (call CommControl.start());
}

/** stop generic communication interface **/
command result_t StdControl.stop(){
  return (call CommControl.stop());
}

/**
 * A module-scoped inline function.
 * Decide whether a received message is new: its sequence number has to be
 * within 127 of the previous sequence number. Also drops the message
 * if the module is still dealing with the previous broadcast.
 **/
inline char is_new_msg(struct SimpleCmdMsg *bmsg) {
  if (bcast_pending) return 0;
  return (((bmsg->seqno - lastSeqno)>0) ||((bmsg->seqno+127)<lastSeqno) ) ;
}

/**
 * A module-scoped inline function. Updates the last sequence number
 * and set the broadcast sending flag.
 **/
inline void remember_msg(struct SimpleCmdMsg *bmsg) {
  lastSeqno = bmsg->seqno;
  bcast_pending = 1;
}

/**
 * Handles the AM type 8 receiving event signaled from ReceiveMsg.
 * Checks if this is a new message and calls ProcessCmd.execute()
 * if so.
 * @return A TOS_MsgPtr.
 **/
event TOS_MsgPtr ReceiveCmdMsg.receive(TOS_MsgPtr pmsg){
  TOS_MsgPtr ret = msg;
  result_t retval;
  struct SimpleCmdMsg *data= (struct SimpleCmdMsg *)pmsg->data;

  // Check if this is a new broadcast message
  //call Leds.greenToggle();
  if (is_new_msg(data)) {
    remember_msg(data);
    retval = call ProcessCmd.execute(pmsg) ;

    // Return a message buffer to the lower levels, and hold on to the
    // current buffer
    ret = msg;
    msg = pmsg;

  }
  return ret;
}

/**
 * Handles the ProcessCmd.done event signaled by ProcessCmd.
 * Once command execution has completed, forward the message.
 * @return Always returns <code>SUCCESS</code>
 **/
event result_t ProcessCmd.done(TOS_MsgPtr pmsg, result_t status) {
```

```
    msg = pmsg;
    //call Leds.redToggle();
    if (status) {
      post forwarder();
    } else {
      bcast_pending = 0;
    }
    return SUCCESS;
  }
} // end of implementation
```

ProcessCmd.nc

```
// $Id: ProcessCmd.nc,v 1.2.14.2 2003/08/18 22:09:36 cssharp Exp $

/*
 * "Copyright (c) 2000-2003 The Regents of the University  of California.
 * All rights reserved.
 *
 * Permission to use, copy, modify, and distribute this software and its
 * documentation for any purpose, without fee, and without written agreement is
 * hereby granted, provided that the above copyright notice, the following
 * two paragraphs and the author appear in all copies of this software.
 *
 * IN NO EVENT SHALL THE UNIVERSITY OF CALIFORNIA BE LIABLE TO ANY PARTY FOR
 * DIRECT, INDIRECT, SPECIAL, INCIDENTAL, OR CONSEQUENTIAL DAMAGES ARISING OUT
 * OF THE USE OF THIS SOFTWARE AND ITS DOCUMENTATION, EVEN IF THE UNIVERSITY OF
 * CALIFORNIA HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
 *
 * THE UNIVERSITY OF CALIFORNIA SPECIFICALLY DISCLAIMS ANY WARRANTIES,
 * INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY
 * AND FITNESS FOR A PARTICULAR PURPOSE.  THE SOFTWARE PROVIDED HEREUNDER IS
 * ON AN "AS IS" BASIS, AND THE UNIVERSITY OF CALIFORNIA HAS NO OBLIGATION TO
 * PROVIDE MAINTENANCE, SUPPORT, UPDATES, ENHANCEMENTS, OR MODIFICATIONS."
 *
 * Copyright (c) 2002-2003 Intel Corporation
 * All rights reserved.
 *
 * This file is distributed under the terms in the attached INTEL-LICENSE
 * file. If you do not find these files, copies can be found by writing to
 * Intel Research Berkeley, 2150 Shattuck Avenue, Suite 1300, Berkeley, CA,
 * 94704.  Attention:  Intel License Inquiry.
 */

includes AM;

/**
 * This interface process a
 * command and is capable of the hnadling of command
 * led_on, led_off, radio_louder and radio_quieter
 */
interface ProcessCmd
{
  /**
   * This command  extracts the command from the message 'pmsg' and
   * executes the command.
   * @return Command execution result.
   */
  command result_t execute(TOS_MsgPtr pmsg);

  /**
   * Indicate that the command contained in 'pmsg' has finished executing.
   * @param status The status of the command completion.
   * @return Always returns SUCCESS.
   */
  event result_t done(TOS_MsgPtr pmsg, result_t status);
}
```

SimpleCmd.nc

```
// $Id: SimpleCmd.nc,v 1.2.14.2 2003/08/18 22:09:36 cssharp Exp $

/*
 * "Copyright (c) 2000-2003 The Regents of the University  of California.
 * All rights reserved.
 *
 * Permission to use, copy, modify, and distribute this software and its
 * documentation for any purpose, without fee, and without written agreement is
 * hereby granted, provided that the above copyright notice, the following
 * two paragraphs and the author appear in all copies of this software.
 *
 * IN NO EVENT SHALL THE UNIVERSITY OF CALIFORNIA BE LIABLE TO ANY PARTY FOR
 * DIRECT, INDIRECT, SPECIAL, INCIDENTAL, OR CONSEQUENTIAL DAMAGES ARISING OUT
 * OF THE USE OF THIS SOFTWARE AND ITS DOCUMENTATION, EVEN IF THE UNIVERSITY OF
 * CALIFORNIA HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
 *
 * THE UNIVERSITY OF CALIFORNIA SPECIFICALLY DISCLAIMS ANY WARRANTIES,
 * INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY
 * AND FITNESS FOR A PARTICULAR PURPOSE.  THE SOFTWARE PROVIDED HEREUNDER IS
 * ON AN "AS IS" BASIS, AND THE UNIVERSITY OF CALIFORNIA HAS NO OBLIGATION TO
 * PROVIDE MAINTENANCE, SUPPORT, UPDATES, ENHANCEMENTS, OR MODIFICATIONS."
 *
 * Copyright (c) 2002-2003 Intel Corporation
 * All rights reserved.
 *
 * This file is distributed under the terms in the attached INTEL-LICENSE
 * file. If you do not find these files, copies can be found by writing to
 * Intel Research Berkeley, 2150 Shattuck Avenue, Suite 1300, Berkeley, CA,
 * 94704.  Attention:  Intel License Inquiry.
 */
/**
 * SimpleCmd is a TinyOS configuration module.
 * It defines the wiring used by SimpleCmdM module.
 * It is wired to Main module's StdControl interface.
 * It components GenericComm's CommControl and ReceiveMsg interfaces
 * to receive AM_SIMPLECMDMSG from base station. It excecutes
 * a command using either Pot or Leds interface depending as to
 * the command type.
 */

includes SimpleCmdMsg;

configuration SimpleCmd {
  provides interface ProcessCmd;
}
implementation {
  components Main, SimpleCmdM, GenericComm as Comm, PotC, LedsC;

  Main.StdControl -> SimpleCmdM;
  SimpleCmdM.Leds -> LedsC;
  ProcessCmd = SimpleCmdM.ProcessCmd;
  SimpleCmdM.CommControl -> Comm;
  SimpleCmdM.ReceiveCmdMsg -> Comm.ReceiveMsg[AM_SIMPLECMDMSG];

  SimpleCmdM.Pot -> PotC;
}
```

SimpleCmdM.nc

```
// $Id: SimpleCmdM.nc,v 1.2.14.3 2003/08/26 09:08:06 cssharp Exp $

/*
 * "Copyright (c) 2000-2003 The Regents of the University  of California.
 * All rights reserved.
 *
 * Permission to use, copy, modify, and distribute this software and its
 * documentation for any purpose, without fee, and without written agreement is
 * hereby granted, provided that the above copyright notice, the following
 * two paragraphs and the author appear in all copies of this software.
 *
 * IN NO EVENT SHALL THE UNIVERSITY OF CALIFORNIA BE LIABLE TO ANY PARTY FOR
 * DIRECT, INDIRECT, SPECIAL, INCIDENTAL, OR CONSEQUENTIAL DAMAGES ARISING OUT
 * OF THE USE OF THIS SOFTWARE AND ITS DOCUMENTATION, EVEN IF THE UNIVERSITY OF
 * CALIFORNIA HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
 *
 * THE UNIVERSITY OF CALIFORNIA SPECIFICALLY DISCLAIMS ANY WARRANTIES,
 * INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY
 * AND FITNESS FOR A PARTICULAR PURPOSE.  THE SOFTWARE PROVIDED HEREUNDER IS
 * ON AN "AS IS" BASIS, AND THE UNIVERSITY OF CALIFORNIA HAS NO OBLIGATION TO
 * PROVIDE MAINTENANCE, SUPPORT, UPDATES, ENHANCEMENTS, OR MODIFICATIONS."
 *
 * Copyright (c) 2002-2003 Intel Corporation
 * All rights reserved.
 *
 * This file is distributed under the terms in the attached INTEL-LICENSE
 * file. If you do not find these files, copies can be found by writing to
 * Intel Research Berkeley, 2150 Shattuck Avenue, Suite 1300, Berkeley, CA,
 * 94704.  Attention:  Intel License Inquiry.
 */
/*
 * Author:  Robert Szewczyk,  Su Ping
 *
 * $\Id$
 */

/**
 * SimpleCmdM is a tiny OS application module.
 * This module demonstrates a simple command interpreter for the TinyOS
 * tutorial. The module receives a command message from the radio, which
 * is passed to the ProcessCmd interface for processing. A task is posted
 * to process the command. The command packet contains a one-byte
 * 'action' field specifying which action to take; as a simple version,
 * this module can only interpret the follwoing commands:
 * Led_on (action = 1), Led_off (2), radio_quieter (3), and radio_louder (4).
 *
 * This module also implements the ProcessCmd interface.
 * @author Robert Szewczyk
 * @author Su Ping
 **/
includes SimpleCmdMsg;

module SimpleCmdM {
  provides          {
    interface StdControl;
    interface ProcessCmd;
  }

  uses {
    interface Leds;
    interface Pot;
    interface ReceiveMsg as ReceiveCmdMsg;
    interface StdControl as CommControl;
  }
}

/*
 *  Module Implementation
 */

implementation
{

  // module scoped variables
  TOS_MsgPtr msg;
  TOS_MsgPtr imp_msg;
  int8_t pending;
  TOS_Msg buf;
  int8_t imp_msg_recvd;
  int8_t sent_back;          //0 means not sent back yet   1 means sent back
```

```
  /**
   *
   * This task evaluates a command and execute it if it is a supported
   * command.The protocol for the command interpreter is that
   * it operates on the message and returns a (potentially modified)
   * message to the calling layer, as well a status word for whether
   * the message should be futher processed.
   *
   * @return Return: None
   **/

  task void cmdInterpret() {
    struct SimpleCmdMsg * cmd = (struct SimpleCmdMsg *) msg->data;
    // do local packet modifications: update the hop count and packet source
    cmd->hop_count++;
    cmd->source = TOS_LOCAL_ADDRESS;

    // Interpret the command: Display the level on red and green led
    //this tells whether msg was received during Bcast from other nodes or from TOSBase

    if (cmd->hop_count & 0x1)
      call Leds.greenOn();
    else
      call Leds.greenOff();
    if (cmd->hop_count & 0x2)
      call Leds.redOn();
    else
      call Leds.redOff();


    // Execute the command
    switch (cmd->action) {
      case LED_ON:
          call Leds.yellowOn();
          break;
      case LED_OFF:
          call Leds.yellowOff();
          break;
      case RADIO_QUIETER:
          call Pot.increase();
          break;
      case RADIO_LOUDER:
          call Pot.decrease();
          break;
      case FORWARD_MSG:
          call Leds.redOn();
          call Leds.yellowOn();
          call Leds.greenOn();
          imp_msg_recvd = 1;
          imp_msg = msg;
          break;
      case RETRIEVE_MSG:
          call Leds.redOff();
          call Leds.yellowOff();
          call Leds.greenOff();
          sent_back = 1;                              //going to send back
          break;
    }
    pending =0;
    if(imp_msg_recvd == 1 && sent_back == 1){         //no impt msg received yet
          imp_msg_recvd = 0;
          sent_back = 0;
          signal ProcessCmd.done(imp_msg, SUCCESS);
          }
    else                                    //imp msg was previously received ---> forward that
          signal ProcessCmd.done(msg, SUCCESS);
  }

  /**
   * Initialization for the application:
   *  1. Initialize module static variables
   *  2. Initialize communication layer
   *  @return Returns <code>SUCCESS</code> or <code>FAILED</code>
   **/

  command result_t StdControl.init() {
    msg = &buf;
    pending = 0;
    imp_msg_recvd = 0;                      //havent received impt msg yet
    sent_back = 0;
    return (call CommControl.init());
  }

/** start communication layer **/
  command result_t StdControl.start(){
```

```
    return (call CommControl.start());
  }
/** stop communication layer **/
  command result_t StdControl.stop(){
    return (call CommControl.stop());
  }

  /**
   * Posts the cmdInterpret() task to handle the recieved command.
   * @return Always returns <code>SUCCESS</code>
   **/
  command result_t ProcessCmd.execute(TOS_MsgPtr pmsg) {
    pending =1;
    msg = pmsg;
    post cmdInterpret();
    return SUCCESS;
  }

  /**
   * Called upon message reception and invokes the ProcessCmd.execute()
   * command.
   * @return Returns a pointer to a TOS_Msg buffer
   **/
  event TOS_MsgPtr ReceiveCmdMsg.receive(TOS_MsgPtr pmsg){
    TOS_MsgPtr ret = msg;
    result_t retval;
    //call Leds.greenToggle();
    retval = call ProcessCmd.execute(pmsg) ;
    if (retval==SUCCESS) {
      return ret;
    } else {
      return pmsg;
    }
  }


  /**
   * Called upon completion of command execution.
   * @return Always returns <code>SUCCESS</code>
   **/
  default event result_t ProcessCmd.done(TOS_MsgPtr pmsg, result_t status) {
    return status;
  }

} // end of implementation
```

SimpleCmdMsg.h

```
// $Id: SimpleCmdMsg.h,v 1.1.2.3 2003/08/23 19:48:39 hohltb Exp $

/*
 * "Copyright (c) 2000-2003 The Regents of the University  of California.
 * All rights reserved.
 *
 * Permission to use, copy, modify, and distribute this software and its
 * documentation for any purpose, without fee, and without written agreement is
 * hereby granted, provided that the above copyright notice, the following
 * two paragraphs and the author appear in all copies of this software.
 *
 * IN NO EVENT SHALL THE UNIVERSITY OF CALIFORNIA BE LIABLE TO ANY PARTY FOR
 * DIRECT, INDIRECT, SPECIAL, INCIDENTAL, OR CONSEQUENTIAL DAMAGES ARISING OUT
 * OF THE USE OF THIS SOFTWARE AND ITS DOCUMENTATION, EVEN IF THE UNIVERSITY OF
 * CALIFORNIA HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
 *
 * THE UNIVERSITY OF CALIFORNIA SPECIFICALLY DISCLAIMS ANY WARRANTIES,
 * INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY
 * AND FITNESS FOR A PARTICULAR PURPOSE.  THE SOFTWARE PROVIDED HEREUNDER IS
 * ON AN "AS IS" BASIS, AND THE UNIVERSITY OF CALIFORNIA HAS NO OBLIGATION TO
 * PROVIDE MAINTENANCE, SUPPORT, UPDATES, ENHANCEMENTS, OR MODIFICATIONS."
 *
 * Copyright (c) 2002-2003 Intel Corporation
 * All rights reserved.
 *
 * This file is distributed under the terms in the attached INTEL-LICENSE
 * file. If you do not find these files, copies can be found by writing to
 * Intel Research Berkeley, 2150 Shattuck Avenue, Suite 1300, Berkeley, CA,
 * 94704.  Attention:  Intel License Inquiry.
 */

/*
 * File Name: SimpleCmd.h
 *
 * Description:
 * This header file defines the AM_SIMPLECMDMSG and AM_LOGMSG message
 * types for the SimpleCmd and SenseLightToLog applications.
 */

enum {
 AM_SIMPLECMDMSG = 8,
 //AM_LOGMSG=9
};

enum {
  LED_ON = 1,
  LED_OFF = 2,
  RADIO_LOUDER  = 3,
  RADIO_QUIETER = 4,
  FORWARD_MSG = 7,                    //added
  RETRIEVE_MSG      = 8               //added
};

typedef struct {
    int nsamples;
    uint32_t interval;
} start_sense_args;

typedef struct {
    uint16_t destaddr;
} read_log_args;

// SimpleCmd message structure
typedef struct SimpleCmdMsg {
    int8_t seqno;
    int8_t action;
    uint16_t source;
    uint8_t hop_count;
    union {
      start_sense_args ss_args;
      read_log_args rl_args;
      uint8_t untyped_args[0];
    } args;
} SimpleCmdMsg;
```

Code on BS_2 is the TinyOS application TOSbase that can be found at http://www.tinyos.net/tinyos-1.x/apps/TOSBase/

Code for Remote Java Server

```
import java.io.*;
import java.net.*;

class TCPServer
{
	public static void main(String argv[]) throws Exception
	{
		String clientSentence, capitalizedSentence;

		ServerSocket welcomeSocket;

		InetAddress local = InetAddress.getLocalHost();

		String ip = local.getHostAddress();

		System.out.println("Server has IP address: "+ ip +"\t nd port: "+welcomeSocket.getLocalPort());

		System.out.println("Waiting for request from client.....");

		Socket connectionSocket = welcomeSocket.accept();

		System.out.println("accepted connection\n");

		BufferedReader inFromClient = new BufferedReader(new InputStreamReader
(connectionSocket.getInputStream()));

		DataOutputStream outToClient = new DataOutputStream(connectionSocket.getOutputStream());

		clientSentence = inFromClient.readLine();

		System.out.println("received client message:\t"+clientSentence);
		welcomeSocket.close();

	}//end of main

}//end of class TCPServer
```

Code on the Responder cell phone

1. Contains: TCP_ClientMIDlet and TCP_Client

TCP_ClientMIDlet.java

```
import javax.microedition.lcdui.*;
import javax.microedition.midlet.*;
import javax.microedition.io.*;
import java.io.*;

public class TCP_ClientMIDlet extends MIDlet implements CommandListener {

  Display display = null;

   // form fields
   TextField toField = null;
   TextField msgField = null;

  Form form;

   static final Command sendCommand = new Command("send", Command.OK, 2);
   static final Command clearCommand = new Command("clear", Command.STOP, 3);

   String to;
   String msg;

   public TCP_ClientMIDlet() {

          display = Display.getDisplay(this);
      form = new Form("Server Address and Message");
      toField = new TextField("Server Address:", "", 25, TextField.ANY);
      msgField = new TextField("Message:", "", 90, TextField.ANY);
   }

   public void startApp() throws MIDletStateChangeException {

      form.append(toField);
      form.append(msgField);

      form.addCommand(clearCommand);
      form.addCommand(sendCommand);
      form.setCommandListener(this);
      display.setCurrent(form);
   }

   public void pauseApp() {
   }

   public void destroyApp(boolean unconditional) {
      notifyDestroyed();
   }

   public void commandAction(Command c, Displayable d) {
      String label = c.getLabel();
      if(label.equals("clear")) {
         destroyApp(true);
      }
      else if (label.equals("send")) {

           to = toField.getString();
         msg = msgField.getString();
         TCP_Client client = new TCP_Client(this,to, msg);

         client.start();

      }//end of if-else

   }//end of method commandAction

}//end of class TCP_ClientMIDlet
```

TCP_Client.java

```
import javax.microedition.midlet.*;
import javax.microedition.io.*;
import javax.microedition.lcdui.*;
import java.io.*;
import java.util.*;

public class TCP_Client implements Runnable {

   private TCP_ClientMIDlet parent;
   private Display display;
   private Form f;
   private StringItem si;
   private StringItem si1;
   private SocketConnection sc;
   private InputStream is;
   private OutputStream os;
   private String serverAddress;
   private String to, msg;
   static final Command exitCommand = new Command("exit", Command.STOP, 3);

   public TCP_Client (TCP_ClientMIDlet m, String to, String msg) {

      parent = m;

      this.to = to;
      this.msg = msg;
      serverAddress = to;

      display = Display.getDisplay(parent);

      f = new Form("TCP Client");
      si1 = new StringItem("Connecting to:", "\n"+serverAddress+" with message: "+msg);
      si = new StringItem("Status:" , " ");

      f.append(si1);
      f.append(si);
      display.setCurrent(f);
   }

   public void start() {
      Thread t = new Thread(this);
      t.start();
   }

   public void run() {

      try {
         si.setText("Opening connection...");
         sc = (SocketConnection) Connector.open("socket://"+serverAddress+":9999");
          //port 9999 is open as per TCPServer programs on Linux in ITIA
         si.setText("Opened connection...");
         is = sc.openInputStream();
         si.setText("Opened input stream...");
         os = sc.openOutputStream();
         si.setText("Opened output stream....and writing bytes...");
         os.write((msg+"\r\n").getBytes()); // message body
         si.setText("Sent Message");
      } catch(IOException e) {
         Alert a = new Alert
         ("TCP_Client", "Cannot connect to server" + serverAddress + "\nPing the server to make sure it is running...",
null, AlertType.ERROR);
         a.setTimeout(Alert.FOREVER);
         display.setCurrent(a);
      } finally {
         try {
            if(is != null) {
               is.close();
            }
            if(os != null) {
               os.close();
            }
            if(sc != null) {
               sc.close();
            }
         } catch(IOException e) {
            e.printStackTrace();
         }
      }
   }

   public void commandAction(Command c, Displayable s) {
      if (c == Alert.DISMISS_COMMAND) {
```

```
      parent.notifyDestroyed();
      parent.destroyApp(true);
   }
}}//end of class
```

Appendix C

ILP Formulation Code

ILP Formulation code contains the following classes: Process.java, ProcessInput.java, LpSolveInput.java and LagrangeRelaxations.java.

Process.java

```
/*****************************************************************************************************************
                                        Driver class        - for preprocessing for lp solve
                                                            - also finds lagrange multipliers
*****************************************************************************************************************/

import lpsolve.*;

public class Process
{
        static String inputFileName;
        static String debugFileName;
        static String lp_solveOutputFileName;
        static String lr_outputFileName;

        public static void main(String args[])
        {

                //scenario 2
                inputFileName = "b_6n_3s.txt";
                debugFileName = "b_6n_3s_debug.txt";
                lp_solveOutputFileName = "b_6n_3s_lp.lp";
                lr_outputFileName = "b_6n_3s_lr.txt";


                //process input

                long startProcessInput = System.currentTimeMillis();

                ProcessInput pi = new ProcessInput(inputFileName, debugFileName);
                pi.start_processing();

                long elapsedTimeMillis_processInput = System.currentTimeMillis()-startProcessInput;
                System.out.println("\n\nTime elapsed(s) to process input: " + (float)elapsedTimeMillis_processInput
/1000F);


                //ILP formulation

                long startLpConstraint = System.currentTimeMillis();

                LpSolveInput lpi = new LpSolveInput(pi, lp_solveOutputFileName);

                lpi.lp_constraintFormulation();

                long elapsedTimeMillis_LpConstraint = System.currentTimeMillis()-startLpConstraint;
                System.out.println("\n\nTime elapsed(s) to write constraints to lp file: " +
(float)elapsedTimeMillis_LpConstraint/1000F);


                //Lagrange Relaxations

                long startLR = System.currentTimeMillis();

                LagrangeRelaxations lr = new LagrangeRelaxations(lp_solveOutputFileName, lr_outputFileName, pi,
lpi);
                lr.beginLagrangeRelaxations();

                long elapsedTimeMillis_LR = System.currentTimeMillis()-startLR;
                System.out.println("\n\nTime elapsed(s) for lagrange relaxations: " +
(float)elapsedTimeMillis_LR/1000F);

        }//end of main

}//end of class Process
```

ProcessInput.java

```
/*************************************************************************************************************************
                                        This class reads the input and initializes matrices R, L, C, T, vector b, and
scalar d
                                        as defined in SLP problem description
*************************************************************************************************************************/

import java.io.*;

class ProcessInput
{
          static String delayConst = "DelayTable.txt";

          static String inputFileName;
          static String debugFileName;
          static int pathCount = 0;

          /********************************************** NETWORK  GIVEN
******************************************************/
          public static int n, s, p, e, qInterval;
          public static int L[][];              //path-link matrix {0,1}                          p by e dimensional
          public static int b[];                          //link bandwidth vector {integers}      e dimensional
          public static int C[][];              //cost matrix {integers}                          n by s dimensional
          public static int d;                            //discount {double/integer}
          public static int R[][][];            //routing matrix {0,1}                                     n by n by
p dimensional
          public static int T[][];              //throughput matrix {integers}                    n by s dimensional
          public static int D[][];              //delay req. matrix                                        n by s
          public static int Q[][];              //delay lookup table                                       10 by 21
for bandwidth 100-1000 (in intervals of 50)
          public static int numRowDelayTable = 10;
          public static int numColDelayTable = 21;
          public static int bandwidthInterval = 100;

          /***************************************************************************************************************
*************/

          public ProcessInput(String inputFile, String debugFile)
          {
                    inputFileName = inputFile;
                    debugFileName = debugFile;

          }//overloadd constructor

          public static void start_processing()
          {
                    System.out.println("processing network information given in file: " + inputFileName);

                    readFile();

                    readDelayConstants();

                    inputEcho();


          }//end of start_processing

          public static void readDelayConstants()
          {
                    try
                    {

                              BufferedReader br = new BufferedReader(new FileReader(delayConst));

                              String sin;

                              //initialize Q
                              Q = new int[numRowDelayTable][numColDelayTable];

                              for(int i = 0; i < numRowDelayTable; i++)
                                        for(int j = 0; j < numColDelayTable; j++)
                                                  Q[i][j] = 0;

                              while((sin = br.readLine()) != null)
                              {

                                        String[] sub = sin.split(" = ");

                                        if(sub[0].equals("max load"))
                                        {
```

```
                                        setQ(sub[1], sub[2]);                                //make
setQ robust enough
                                }
                                else if(sub[0].equals("q"))
                                        qInterval = Integer.parseInt(sub[1]);
                                else
                                        System.out.println(sub[0] + " not valid input in inputfile");

                        }//end of while

                        br.close();

                }catch(IOException ioe)
                {
                        System.out.println("error in method readDelayConstants in class ProcessInput: "+
ioe.getMessage());
                }//end of ioexception catch

        }//end of readDelayConstants

        public static void readFile()
        {
                try
                {

                        BufferedReader br = new BufferedReader(new FileReader(inputFileName));

                        String sin;

                        while((sin = br.readLine()) != null)
                        {

                                String[] sub = sin.split(" = ");

                                if(sub[0].equals("n"))
                                        n = Integer.parseInt(sub[1]);
                                else if(sub[0].equals("s"))
                                        s = Integer.parseInt(sub[1]);
                                else if(sub[0].equals("p"))
                                        p = Integer.parseInt(sub[1]);
                                else if(sub[0].equals("e"))
                                        e = Integer.parseInt(sub[1]);
                                else if(sub[0].equals("L"))
                                        //set pathlink matrix
                                        setL(sub[1]);
                                else if(sub[0].equals("b"))
                                        //set bandwidth vector
                                        setb(sub[1]);
                                else if(sub[0].equals("C"))
                                        //set cost matrix
                                        setC(sub[1]);
                                else if(sub[0].equals("d"))
                                        d = Integer.parseInt(sub[1]);
                                else if(sub[0].equals("R"))
                                        //set routing matrix
                                        setR(sub[1]);
                                else if(sub[0].equals("T"))
                                        //set throughput
                                        setT(sub[1]);
                                else if(sub[0].equals("D"))
                                        setD(sub[1]);
                                else
                                        System.out.println(sub[0] + " not valid input in inputfile");

                        }//end of while

                        //once all matrices have been initialized update the number of paths, and edges
                        p = (2*p) + n;

                        //keep track of org e, i.e. w/o loops
//                      org_e = e;
                        e = e + n;

                        br.close();

                }catch(IOException ioe)
                {
                        System.out.println("error in method readFile in class ProcessInput: "+ ioe.getMessage());
                }//end of ioexception catch

        }//end of readFile()

        public static void setT(String sin)
        {
                T = new int[n][s];
```

```
		for(int i = 0; i < n; i++)
			for(int j = 0; j < s; j++)
				T[i][j] = 0;

		String[] row = sin.split(";");

		for(int i = 0; i < row.length; i++)
		{
			String[] cost = row[i].split(",");

			for(int j = 0; j < cost.length; j++)
				T[i][j] = Integer.parseInt(cost[j]);

		}//end of for

		//print2DMatrix(System.out, T, n, s);

	}//end of setT

	public static void setD(String sin)
	{
		D = new int[n][s];

		for(int i = 0; i < n; i++)
			for(int j = 0; j < s; j++)
				D[i][j] = 0;

		String[] row = sin.split(";");

		for(int i = 0; i < row.length; i++)
		{
			String[] cost = row[i].split(",");

			for(int j = 0; j < cost.length; j++)
				D[i][j] = Integer.parseInt(cost[j]);

		}//end of for

		//print2DMatrix(System.out, D, n, s);

	}//end of setD


	public static void setC(String sin)
	{
		C = new int[n][s];

		for(int i = 0; i < n; i++)
			for(int j = 0; j < s; j++)
				C[i][j] = 0;

		String[] row = sin.split(";");

		for(int i = 0; i < row.length; i++)
		{
			String[] cost = row[i].split(",");

			for(int j = 0; j < cost.length; j++)
				C[i][j] = Integer.parseInt(cost[j]);

		}//end of for

		//print2DMatrix(System.out, C, n, s);

	}//end of setC

	public static void setb(String sin)
	{
		b = new int[e+n];

		for (int i = 0; i < e+n; i++)
			b[i] = 0;

		String[] bndwd = sin.split(",");

		for(int i = 0; i < bndwd.length; i++)
			b[i] = Integer.parseInt(bndwd[i]);

		//zero cycle loops for each node
		for(int i = e; i < e+n; i++)
			b[i] = 1000;

		//printVector(System.out, b, e);
```

```
}//end of setb

public static void setQ(String r, String sin)
{
          int row = ((Integer.parseInt(r))/bandwidthInterval)-1;

          String[] queDelay = sin.split(",");

          for(int i = 0; i < queDelay.length; i++)
                    Q[row][i] = Integer.parseInt(queDelay[i]);

}//end of overloaded setQ

public static void setR(String sin)
{
          int numCol = 2*p+n;

          R = new int[n][n][numCol];

          for(int i = 0; i < n; i++)
                    for(int j = 0; j < n; j++)
                              for(int k = 0; k < numCol; k++)
                                        R[i][j][k] = 0;

          String[] chunks = sin.split(";");

          int pathCount = p;

          for(int i = 0; i < chunks.length; i++)
          {
                    String[] pair = chunks[i].split(":");

                    //getting row and col indices
                    String[] r_c = pair[0].split("-");
                    int row = Integer.parseInt(r_c[0]);
                    int col = Integer.parseInt(r_c[1]);

                    //getting paths used
                    String[] paths = pair[1].split(",");

                    for(int k = 0; k < paths.length; k++)
                    {
                              R[row-1][col-1][Integer.parseInt(paths[k])-1] = 1;
                              R[col-1][row-1][pathCount++] = 1;
                    }

          }//end of i

          //add the zero cycle loops on each node
          for(int i = 0, m = pathCount; i < n; i++, m++)
                              R[i][i][m] = 1;

}//end of setR

public static void setL(String sin)
{
          L = new int[(2*p)+n][e+n];

          for(int i = 0; i < (2*p)+n; i++)
                    for(int j = 0; j < e+n; j++)
                              L[i][j] = 0;

          String[] row = sin.split(";");

          for(int i = 0,LpathCount = p; i < row.length; i++,LpathCount++)
          {
                    String[] edges = row[i].split(",");

                    for(int j = 0; j < edges.length; j++)
                    {
                              L[i][Integer.parseInt(edges[j])-1]=1;
                              L[LpathCount][Integer.parseInt(edges[j])-1] = 1;
                    }

          }//end of for

          for(int i = (2*p), j = e; i < (2*p)+n && j < e+n; i++, j++)
                    L[i][j] = 1;

          //print2DMatrix(System.out, L, p, e);

}//end of setL

public static void printVector(PrintStream w, int M[], int l)
{
```

```
		w.println("= {");

		for(int i = 0; i < l; i++)
		{
			w.print(M[i]);
			if(i != l-1)
				w.print(" ");
			else
				w.println();
		}

		w.println("}");
	}//end of printVector

	public static void printVector(PrintWriter w, int M[], int l)
	{
		w.println("= {");

		for(int i = 0; i < l; i++)
		{
			w.print(M[i]);
			if(i != l-1)
				w.print(" ");
			else
				w.println();
		}

		w.println("}");
	}//end of printVector

	public static void print2DMatrix(PrintStream w, int M[][], int row, int col)
	{
		w.println("= {");

		for(int i = 0; i < row; i++)
		{
			for(int j=0; j < col; j++)
			{
				w.print(M[i][j]);
				if(j != col-1)
					w.print(" ");
				else
					w.println();
			}
		}

		w.println("}");
	}//end of print2DMatrix

	public static void print2DMatrix(PrintWriter w, int M[][], int row, int col)
	{
		w.println("= {");

		for(int i = 0; i < row; i++)
		{
			for(int j=0; j < col; j++)
			{
				w.print(M[i][j]);
				if(j != col-1)
					w.print(" ");
				else
					w.println();
			}

			if((i+1)%10 == 0)
				w.println("\ni = "+i+"\n");
		}

		w.println("}");
	}//end of print2DMatrix

	public static void print3DMatrix(PrintStream w, int M[][][], int row, int col, int pg)
	{
		w.println("= {");

		for(int i = 0; i < row; i++)
		{
			for(int j=0; j < col; j++)
			{
				w.println("\nsrc-dest pair: " + i + "," + j + ":");
```

```
                    for(int k = 0; k < pg; k++)
                    {
                            w.print(M[i][j][k]);

                            if(j != pg-1)
                                    w.print(" ");
                            else
                                    w.println();

                    }//end of k

            }//end of j

        }//end of i

        w.println("}");

}//end of print3DMatrix

public static void print3DMatrix(PrintWriter w, int M[][][], int row, int col, int pg)
{
        w.println("= {");

        for(int i = 0; i < row; i++)
        {
                for(int j=0; j < col; j++)
                {
                        w.print("\nsrc-dest pair: " + i + "," + j + ":\n");

                        for(int k = 0; k < pg; k++)
                        {
                                w.print(M[i][j][k]);
                                if(j != pg-1)
                                        w.print(" ");
                                else
                                w.println();

                        }//end of k

                }//end of j

        }//end of i

        w.println("}");

}//end of print3DMatrix


public static void inputEcho()
{
        try{

                System.out.println("debug file: " + debugFileName);

                PrintWriter pw = new PrintWriter(new FileWriter(debugFileName));

                pw.println(" n = " + n);
                pw.println(" s = " + s);
                pw.println(" p = " + p);
                pw.println(" e = " + e);
                pw.println(" d = " + d);

                pw.println(" L = ");
                print2DMatrix(pw, L, p, e);

                pw.println(" b = ");
                printVector(pw, b, e);

                pw.println(" C = ");
                print2DMatrix(pw, C, n, s);

                pw.println(" R = ");
                print3DMatrix(pw, R, n, n, p);

                pw.println(" T = ");
                print2DMatrix(pw, T, n, s);

                pw.println(" D = ");
                print2DMatrix(pw, D, n, s);

                pw.println(" Q = ");
                print2DMatrix(pw, Q, numRowDelayTable, numColDelayTable);

                pw.println("interval in Q = " + qInterval);
```

```
                    pw.close();

            }catch(IOException ioe)
            {
                    System.out.println("error in inputEcho: "+ioe.getMessage());
            }

    }//end of output_format

    public static int find_bmax()
    {
            int max = b[0];

            for(int i = 1; i < b.length; i++)
                    if(b[i] > max)
                            max = b[i];

            return max;

    }//end of find_bmax

    public static int find_qmax()                          //max delay for any load on any link capacity
    {
            int max = Q[0][0];

            for(int i = 0; i < numRowDelayTable; i++)
                    for(int j = 0; j < numColDelayTable; j++)
                            if(Q[i][j] > max)
                                    max = Q[i][j];

            return max;

    }//end of find_qmax

}//end of ProcessInput
```

LpSolveInput.java

```
import java.io.*;

public class LpSolveInput
{
    static String lpOutputFileName ="";
    static ProcessInput pi;

    static int[][] counters;      //row is constraint # in text file and column# is start of row # in lp model
    static int constrNum;                 //constr num from formulations
    static int rowNum;                    //rowNum in LP model

    public LpSolveInput(ProcessInput _pi, String filename)
    {
            pi = _pi;
            lpOutputFileName = filename;

            initCounters();

    }//end overlaoded constructor

    public static void initCounters()
    {
            counters = new int[40][1];

            constrNum = 0;
            rowNum = 0;

            for(int i = 0; i < counters.length; i++)
                    counters[i][0] = 0;

    }//end of initCounters

    public static void lp_constraintFormulation()
    {
            try{

                    PrintWriter pw = new PrintWriter(new FileWriter(lpOutputFileName));

                    //objective function
                    pw.println(lp_objectiveFunction());

                    counters[constrNum++][0] = rowNum;

                    pw.println();
                    pw.println("\n\n/* service installation cost and location constraints */");
```

```
pw.println("/* constraint 1 */");
//constraint 1
for(int i = 0; i < pi.n; i++)
        pw.println(lp_constraint1(i) + ";");

counters[constrNum++][0] = rowNum;

pw.println();
pw.println("/* constraint 2 */");
//constraint 2
for(int i = 0; i < pi.n; i++)
        pw.println(lp_constraint2(i) + ";");

counters[constrNum++][0] = rowNum;

pw.println();
pw.println("\n\n/* throughput constraints */");
pw.println("/* constraint 3 */");
//constraint 3
for(int j = 0; j < pi.n; j++)
        for(int k = 0; k < pi.s; k++)
                pw.print(lp_constraint3(j,k));

counters[constrNum++][0] = rowNum;

pw.println();
pw.println("/* constraint 4 */");
//constraint 4
for(int i = 0; i < pi.n; i++)
        for(int k = 0; k < pi.s; k++)
                pw.println(lp_constraint4(i,k) + ";");

counters[constrNum++][0] = rowNum;

pw.println("/* constraint 5 */");
//constraint 5
for(int i = 0; i < pi.n; i++)
        for(int k = 0; k < pi.s; k++)
                pw.println(lp_constraint5(i,k) + ";");

counters[constrNum++][0] = rowNum;

pw.println();
pw.println("\n\n/* network link capacity constraint */");
pw.println("/* constraint 6 */");
//constraint 6
for(int l = 0; l < pi.e; l++)
        pw.println(lp_constraint6(l));

counters[constrNum++][0] = rowNum;

pw.println("\n\n\n/*********************** DELAY CONSTRAINTS
************************/\n\n");


/***** path service indicator *****/

pw.println();
pw.println("\n/* path serivce indicator */");
pw.println("/* constraint 7 */");
//constraint 7
for(int m = 0; m < pi.p; m++)
        for(int k = 0; k < pi.s; k++)
                pw.println(lp_constraint7(m,k) + ";");

counters[constrNum++][0] = rowNum;

pw.println();
pw.println("/* constraint 8 */");
//constratin 8
for(int m = 0; m < pi.p; m++)
        for(int k = 0; k < pi.s; k++)
                pw.println(lp_constraint8(m,k) + ";");

counters[constrNum++][0] = rowNum;

/***** compute load on link and round it up if necessary *****/

pw.println();
pw.println("\n/* compute load on link and round it up if necessary */");
pw.println("/* constraint 9 */");
//constraint9
//compute load at each link
```

```
for(int l = 0; l < pi.e; l++)
        pw.println(lp_constraint9(l));

counters[constrNum++][0] = rowNum;

pw.println("/* constraint T9 */");
//constraintT9
//compute load at each link
for(int l = 0; l < pi.e; l++)
        pw.println(lp_constraintT9(l));

counters[constrNum++][0] = rowNum;

pw.println();
pw.println("/* constraint 10 */");
//constraint10
for(int l = 0; l < pi.e; l++)
        pw.println(lp_constraint10(l)+";");

counters[constrNum++][0] = rowNum;

pw.println();
pw.println("/* constraint T10 */");
//constraintT10
for(int l = 0; l < pi.e; l++)
        pw.println(lp_constraintT10(l)+";");

counters[constrNum++][0] = rowNum;


/* compute the index that matches the load on a link to the delay in the lookup table */

pw.println("\n/* compute the index that matches the load on a link to the delay in the
lookup table */");

pw.println();
pw.println("/* constraint 11 */");
//constraint11
for(int l = 0; l < pi.e; l++)
        for(int i = 0,j=0; i < pi.numColDelayTable && j <= pi.b[l];
i++,j+=pi.qInterval)
                pw.println(lp_constraint11(l,i) + ";");

counters[constrNum++][0] = rowNum;

pw.println("/* constraint T11 */");
//constraintT11
for(int l = 0; l < pi.e; l++)
        for(int i = 0,j=0; i < pi.numColDelayTable && j <= pi.b[l];
i++,j+=pi.qInterval)
                pw.println(lp_constraintT11(l,i) + ";");

counters[constrNum++][0] = rowNum;

pw.println();
pw.println("/* constraint 12 */");
//constraint12
for(int l = 0; l < pi.e; l++)
        pw.println(lp_constraint12(l) + ";");

counters[constrNum++][0] = rowNum;

pw.println();
pw.println("/* constraint 13 */");
//constraint13
for(int l = 0; l < pi.e; l++)
        for(int i = 0,j=0; i < pi.numColDelayTable && j <= pi.b[l];
i++,j+=pi.qInterval)
                pw.println(lp_constraint13(l,i) + ";");

counters[constrNum++][0] = rowNum;


pw.println();
pw.println("/* constraint T13 */");
//constraintT13
for(int l = 0; l < pi.e; l++)
        for(int i = 0,j=0; i < pi.numColDelayTable && j <= pi.b[l];
i++,j+=pi.qInterval)
                pw.println(lp_constraintT13(l,i) + ";");

counters[constrNum++][0] = rowNum;


/* compute delay on link and then path */
```

```
pw.println("\n/* compute delay on link and then path */");
pw.println();
pw.println("/* constraint 14 */");
//constraint 14
for(int l = 0; l < pi.e; l++)
          pw.println(lp_constraint14(l) + ";");


counters[constrNum++][0] = rowNum;

pw.println();
pw.println("/* constraint T14 */");
//constraint T14
for(int l = 0; l < pi.e; l++)
          pw.println(lp_constraintT14(l) + ";");

counters[constrNum++][0] = rowNum;

pw.println();
pw.println("/* constraint 15 */");
//constraint 15
for(int m = 0; m < pi.p; m++)
          pw.println(lp_constraint15(m) + ";");

counters[constrNum++][0] = rowNum;

pw.println();
pw.println("/* constraint T15 */");
//constraint T15
for(int m = 0; m < pi.p; m++)
          pw.println(lp_constraintT15(m) + ";");

counters[constrNum++][0] = rowNum;



/* meeting delay requirements */

pw.println("\n/* meeting delay requirements */");
pw.println();
pw.println("/* constraint 16 */");
//constraint 16
for(int i = 0; i < pi.n; i++)
          for(int j = 0; j < pi.n; j++)
                    for(int k = 0; k < pi.s; k++)
                              if(pi.T[j][k] > 0)
                              {
                                        for(int m = 0; m < pi.p; m++)
                                                  pw.print(lp_constraint16(i,j,k,m));
                              }

counters[constrNum++][0] = rowNum;

pw.println();
pw.println("/* constraint 17 */");
//constraint 17
for(int i = 0; i < pi.n; i++)
          for(int j = 0; j < pi.n; j++)
                    for(int k = 0; k < pi.s; k++)
                              if(pi.T[j][k] > 0)
                              {
                                        for(int m = 0; m < pi.p; m++)
                                                  pw.print(lp_constraint17(i,j,k,m));
                              }

counters[constrNum++][0] = rowNum;

pw.println();
pw.println("/* constraint 18 */");
//constraint 18
for(int i = 0; i < pi.n; i++)
          for(int j = 0; j < pi.n; j++)
                    for(int k = 0; k < pi.s; k++)
                              if(pi.T[j][k] > 0)
                              {
                                        for(int m = 0; m < pi.p; m++)
                                                  pw.print(lp_constraint18(i,j,k,m));
                              }

counters[constrNum++][0] = rowNum;

pw.println();
pw.println("/* constraint 19 */");
//constraint 19
for(int j = 0; j < pi.n; j++)
```

```
                                for(int k = 0; k < pi.s; k++)
                                        pw.println(lp_constraint19(j,k));

                        counters[constrNum++][0] = rowNum;

                        pw.println();
                        pw.println("/* constraint 20 */");
                        //constraint 20
                        for(int j = 0; j < pi.n; j++)
                                for(int k = 0; k < pi.s; k++)
                                        pw.println(lp_constraint20(j,k));

                        counters[constrNum++][0] = rowNum;

                        /******************************** END OF CCONSTRAINT FORMULATIONS
******************************/

                        //integer linear programming -- lp_solve: declare variables as int
                        pw.println("\n\n" + lp_variableBound());

                        //integer linear programming -- lp_solve: declare variables as int
                        pw.println("\n\n" + lp_variableDeclaration());

                        System.out.println("Input file for lp_solve: " + lpOutputFileName + " is ready for
use.");

                        pw.close();

                }catch(IOException ioe)
                {
                        System.out.println("error in constraintFormulation: "+ioe.getMessage());
                }

        }//end of lp_constraintFormulation

        public static String lp_objectiveFunction()
        {
                String output = "";

                output = "min: ";

                for(int i = 0; i < pi.n; i++)
                        for(int k=0; k < pi.s; k++)
                                output += pi.C[i][k] + " " + "X"+i+"_"+k + " + ";

                for(int i = 0; i < pi.n; i++)
                {
                        output += pi.d + " U"+i;
                        if(i!= pi.n-1)
                                output += " + ";
                        else
                                output += ";";
                }

                rowNum++;

                return output;

        }//end of lp_objectiveFunction

        public static String lp_constraint1(int i)
        {
                String output = "";

                for(int k = 0; k < pi.s; k++)
                {
                        output += "X"+i+"_"+k;
                        if(k!=pi.s-1)
                                output += " + ";
                        else
                                output += " ";

                }

                output += " - " +(pi.find_bmax())+" U"+i +" <= 0" ;

                rowNum++;

                return output;

        }//end of lp_constraint1

        public static String lp_constraint2(int i)
        {
                String output = "";
```

```
		for(int k = 0; k < pi.s; k++)
			output += " - X"+i+"_"+k + " ";

		output += " + U"+i +" <= 0";

		rowNum++;

		return output;

	}//end of lp_constraint2

	public static String lp_constraint3(int j, int k)
	{
		String output = "";

		for(int i = 0; i < pi.n; i++)
			for(int m = 0; m < pi.p; m++)
				if(pi.R[i][j][m]==1)
					output += " - Z"+m+"_"+k + " ";

		output += " <= -" + pi.T[j][k] + ";\n";

		rowNum++;

		return output;

	}//end of lp_constraint3


	public static String lp_constraint4(int i, int k)
	{
		String output = "";

		output += "X"+i+"_"+k;

		for(int j = 0; j < pi.n; j++)
		{
			for(int m = 0; m < pi.p; m++)
			{
				if(pi.R[i][j][m]==1)
					output += " - " + " Z"+m+"_"+k;
			}
		}

		output += " <= 0";

		rowNum++;

		return output;

	}//end of lp_constraint4

	public static String lp_constraint5(int i, int k)
	{
		String output = "";

		output += " - " +(pi.find_qmax()*pi.p) + " X"+i+"_"+k;

		for(int j = 0; j < pi.n; j++)
		{
			for(int m = 0; m < pi.p; m++)
			{
				if(pi.R[i][j][m]==1)
					output += " +  " + " Z"+m+"_"+k;
			}
		}

		output += " <= 0";

		rowNum++;

		return output;

	}//end of lp_constraint5

	public static String lp_constraint6(int l)
	{
		String output = "";
		boolean firstTerm = false;

		for(int m = 0; m < pi.p; m++)
			for(int k = 0; k < pi.s; k++)
			{
				if(pi.L[m][l]==1)
```

```
                    {
                            if(firstTerm == false)
                            {
                                    firstTerm = true;
                                    output += "Z"+m+"_"+k;
                            }
                            else
                                    output += " + Z"+m+"_"+k;
                    }
            }

        if(firstTerm == true)
        {
                output += " <= " + pi.b[l] + ";";
                rowNum++;
        }

        return output;

}//end of lp_constraint6

/*****************************************************************************************************/

//computing Y(m,k) - path service indicator
public static String lp_constraint7(int m, int k)
{
        String output = "";

        output += " - "+(pi.find_qmax()*pi.p) + " Y"+m+"_"+k + " + Z"+m+"_"+k +" <= 0";

        rowNum++;

        return output;

}//end of lp_constraint7

public static String lp_constraint8(int m, int k)
{
        String output = "";

        output += " Y"+m+"_"+k + " - " + " Z"+m+"_"+k + " <= 0";

        rowNum++;

        return output;

}//end of lp_constraint8

/*****************************************************************************************************/

//load vector V
public static String lp_constraint9(int l)
{
        String output = "";

        boolean change = false;

        output += "V"+l+ " ";


        for(int m = 0; m < pi.p; m++)
                for(int k = 0; k < pi.s; k++)
                        if(pi.L[m][l]==1)
                        {
                                output += " - Z"+m+"_"+k;
                                change = true;
                        }

        output += "<= 0;";

        if(change == true)
                rowNum++;

        return output;

}//end of lp_constraint9


//to remove equal sign

public static String lp_constraintT9(int l)
{
        String output = "";
```

```
		boolean change = false;

		output += " - V"+l;

		for(int m = 0; m < pi.p; m++)
			for(int k = 0; k < pi.s; k++)
				if(pi.L[m][l]==1)
				{
					output += " + Z"+m+"_"+k;
					change = true;
				}

		output += "<= 0;";

		if(change == true)
			rowNum++;

		return output;

	}//end of lp_constraintT9

	//constraints related to rounding

	public static String lp_constraint10(int l)
	{
		String output = "";

		output += "V"+l + " - "+ (pi.qInterval)+" a"+l +" <= 0";

		rowNum++;

		return output;

	}//end of constraint10

	public static String lp_constraintT10(int l)
	{
		String output = "";

		output += " - V"+l + " + "+ (pi.qInterval)+" a"+l +" <= 0";

		rowNum++;

		return output;

	}//end of constraintT10

	/****************************************************************************************************/

	//compute index to be looked up in table

	////////042407 version
	public static String lp_constraint11(int l, int i)
	{
		String output = "";

		output += "V" +l + " - " + (pi.qInterval*(i)) + " - " +
(pi.numRowDelayTable*pi.bandwidthInterval+pi.find_qmax()) + " c"+l+"_"+i + " <= 0";

		rowNum++;

		return output;

	}//end of lp_constraint11

	public static String lp_constraintT11(int l, int i)
	{
		String output = "";

		output += " - V" +l + " + " + (pi.qInterval*(i)) + " - " +
(pi.numRowDelayTable*pi.bandwidthInterval+pi.find_qmax()) + " c"+l+"_"+i + " <= 0";

		rowNum++;

		return output;

	}//end of lp_constraintT11

	public static String lp_constraint12(int l)
	{
		String output = "";
```

```
        int i=0,j=0;

        for(i = 0,j=0; i < pi.numColDelayTable && j < pi.b[l]; i++,j+=pi.qInterval)
                output += "c"+l+"_"+i + " + ";

        output += "c"+l+"_"+i + " <= " + pi.b[l]/pi.qInterval;

        rowNum++;

        return output;
}//end of lp_constraint12

public static String lp_constraint13(int l, int i)
{
        String output = "";

        output += "k"+l+"_"+i + " + c"+l+"_"+i+" <= 1";

        rowNum++;

        return output;

}//end of lp_constraint13

public static String lp_constraintT13(int l, int i)
{
        String output = "";

        output += "- k"+l+"_"+i + " - c"+l+"_"+i+" <= -1";

        rowNum++;

        return output;

}//end of lp_constraintT13

/*****************************************************************************************************/

//compute delay on link and path

public static String lp_constraint14(int l)
{
        int row = (pi.b[l]/pi.bandwidthInterval)-1;

        String output = "";

        output += "G" + l;

        for(int j = 0,counter=0; j < pi.numColDelayTable && counter <= pi.b[l]; j++,counter+=pi.qInterval)
                if(pi.Q[row][j] > 0)
                        output += " - "+pi.Q[row][j] + " k" + l+"_"+j;

        output += " <= 0";

        rowNum++;

        return output;

}//end of lp_constraint14

public static String lp_constraintT14(int l)
{
        int row = (pi.b[l]/pi.bandwidthInterval)-1;

        String output = "";

        output += " - G" + l;

        for(int j = 0,counter=0; j < pi.numColDelayTable && counter <= pi.b[l]; j++,counter+=pi.qInterval)
                if(pi.Q[row][j] > 0)
                        output += " + "+pi.Q[row][j] + " k" + l+"_"+j;

        output += " <= 0";

        rowNum++;

        return output;

}//end of lp_constraintT14


public static String lp_constraint15(int m)
{
```

```
        String output ="";

        output += "H"+m;

        for(int l = 0 ; l < pi.e; l++)
                if(pi.L[m][l]==1)
                        output += " - G"+l;

        output += " <= 0";

        rowNum++;

        return output;

}//end of lp_constraint15

public static String lp_constraintT15(int m)
{
        String output ="";

        output += " - H"+m;

        for(int l = 0 ; l < pi.e; l++)
                if(pi.L[m][l]==1)
                        output += " + G"+l;

        output += " <= 0";

        rowNum++;

        return output;

}//end of lp_constraintT15

/*******************************************************************************************/

//meeting delay requirement

public static String lp_constraint16(int i, int j, int k, int m)
{
        String output = "";

        if(pi.R[i][j][m]==1)
        {
                output += " - Y"+m+"_"+k + " - r"+i+"_"+j+"_"+k+"_"+m + " <= -1;\n";
                rowNum++;
        }

        return output;

}//end of lp_constraint16

public static String lp_constraint17(int i, int j, int k, int m)
{
        String output = "";

        if(pi.R[i][j][m]==1)
        {
                output += "Y"+m+"_"+k + " + r"+i+"_"+j+"_"+k+"_"+m + " <= 1;\n";
                rowNum++;
        }

        return output;

}//end of lp_constraint17


public static String lp_constraint18(int i, int j, int k, int m)
{
        String output = "";

        if(pi.R[i][j][m]==1)
        {
                //LHS
                output += "\n H"+m;
                output += " - " + (pi.find_qmax()+pi.find_bmax()) + " r"+i+"_"+j+"_"+k+"_"+m;
                //RHS
                output += " <= " + pi.D[j][k] + ";";

                rowNum++;
        }

        return output;
```

```
}//end of lp_constraint18


public static String lp_constraint19(int j, int k)
{
        String output = "";

        for(int i = 0; i < pi.n; i++)
                for(int m = 0; m < pi.p; m++)
                        if(pi.R[i][j][m]==1)
                                output += " - " + (pi.find_qmax()*pi.p) +" Y"+m+"_"+k;


        output += " <= - " + pi.T[j][k] + ";";
        rowNum++;

        return output;

}//end of lp_constraint19

public static String lp_constraint20(int j, int k)
{
        String output = "";

        for(int i = 0; i < pi.n; i++)
                for(int m = 0; m < pi.p; m++)
                        if(pi.R[i][j][m]==1)
                                output += " + Y"+m+"_"+k;


        output += " <= " + pi.T[j][k] + ";";
        rowNum++;

        return output;

}//end of lp_constraint20


/******************************************************************************************/
/******************************************************************************************/

        public static String lp_variableBound()
        {
                String output = "\n\n";

                for(int i = 0; i < pi.n; i++)
                        for(int k = 0; k < pi.s; k++)
                                output += "0 <= X"+i+"_"+k + " <= 1;\n";

                for(int i = 0; i < pi.n; i++)
                        output += "0 <= U"+i + "<= 1;\n";

                for(int m = 0; m < pi.p; m++)
                        for(int k = 0; k < pi.s; k++)
                                output += "0 <= Y"+m+"_"+k + " <= 1;\n";

                for(int l = 0; l < pi.e; l++)
                        for(int i =0,j=0; i < pi.numColDelayTable && j <= pi.b[l]; i++,j+=pi.qInterval)
                                output += "0 <= c"+l+"_"+i + " <= 1;\n";

                for(int l = 0; l < pi.e; l++)
                        for(int i =0,j=0; i < pi.numColDelayTable && j <= pi.b[l]; i++,j+=pi.qInterval)
                                output += "0 <= k"+l+"_"+i + " <= 1;\n";

                for(int i = 0; i < pi.n; i++)
                        for(int j = 0; j < pi.n; j++)
                                for(int k = 0; k < pi.s; k++)
                                        if(pi.T[j][k] > 0)
                                        {
                                                for(int m = 0; m < pi.p; m++)
                                                        if(pi.R[i][j][m] == 1)
                                                                output += "0 <=
z"+i+"_"+j+"_"+k+"_"+m + " <= 1;\n";
                                        }


                return output;

}//end of variableBound

/******************************************************************************************/
/******************************************************************************************/

public static String lp_variableDeclaration()
```

```
	{
		String output = "\n\n";

		output += "\n\n";

		for(int i = 0; i < pi.n; i++)
			for(int k = 0; k < pi.s; k++)
				output += "int X"+i+"_"+k + ";\n";

		for(int i = 0; i < pi.n; i++)
			output += "int U"+i + ";\n";

		for(int i = 0; i < pi.e; i++)
			output += "int V"+i + ";\n";

		for(int l = 0; l < pi.e; l++)
				output += "int G"+l+ ";\n";

		for(int m = 0; m < pi.p; m++)
			output += "int H"+m + ";\n";

		for(int m = 0; m < pi.p; m++)
			for(int k = 0; k < pi.s; k++)
				output += "int Y"+m+"_"+k + ";\n";

		for(int i = 0; i < pi.e; i++)
			output += "int a"+i + ";\n";

		for(int l = 0; l < pi.e; l++)
			for(int i =0,j=0; i < pi.numColDelayTable && j <= pi.b[l]; i++,j+=pi.qInterval)
				output += "int c"+l+"_"+i + ";\n";

		for(int l = 0; l < pi.e; l++)
			for(int i =0,j=0; i < pi.numColDelayTable && j <= pi.b[l]; i++,j+=pi.qInterval)
				output += "int k"+l+"_"+i + ";\n";

		for(int i = 0; i < pi.n; i++)
			for(int j = 0; j < pi.n; j++)
				for(int k = 0; k < pi.s; k++)
					if(pi.T[j][k] > 0)
					{
						for(int m = 0; m < pi.p; m++)
							if(pi.R[i][j][m] == 1)
								output += "int r"+i+"_"+j+"_"+k+"_"+m
+";\n";
					}

		for(int m = 0; m < pi.p; m++)
			for(int k = 0; k < pi.s; k++)
			{
				output += "int Z"+m+"_"+k;

				if(m==(pi.p-1) && (k==pi.s-1))
					output += ";\n";
				else
					output += ";\n";
			}

		return output;

	}//end of variableDeclaration

}//end of LpSolveInput
```

The sample delay table input file.

```
q = 50
max load = 100 = 0,200,500
max load = 200 = 0,67,100,200,500
max load = 300 = 0,40,50,67,100,200,500
max load = 400 = 0,29,33,40,50,67,100,200,500
max load = 500 = 0,22,25,29,33,40,50,67,100,200,500
max load = 600 = 0,18,20,22,25,29,33,40,50,67,100,200,500
max load = 700 = 0,15,16,18,20,22,25,29,33,40,50,67,100,200,500
max load = 800 = 0,13,14,15,16,18,20,22,25,29,33,40,50,67,100,200,500
max load = 900 =
0,11,12,13,14,15,16,18,20,22,25,29,33,40,50,67,100,200,500
max load = 1000 =
0,9,10,11,12,13,14,15,16,18,20,22,25,29,33,40,50,67,100,200,500
```

Appendix D

Lagrangean Relaxation Code

LagrangeRelaxations.java

```java
/*
        */

import java.io.*;
import lpsolve.*;


public class LagrangeRelaxations
{
        LpSolve orgLpModel;
        LpSolve lr_lps;

        ProcessInput pi;
        LpSolveInput lpi;

        double lrMult[];

        int ignoreConstr[];
        int numConstrNotRelaxed;

        String outputFileName;
        String lrOutputFile;

        public LagrangeRelaxations(String LpFileName, String _outputFileName, ProcessInput _pi, LpSolveInput _lpi)
        {
                outputFileName = _outputFileName;
                lrOutputFile = _outputFileName;
                pi = _pi;
                lpi = _lpi;

                //just picking up the ilp model from the test file as a matrix.
                try
                {
                        //initiatlize lp mode
                        orgLpModel = LpSolve.makeLp(0,0);
                        orgLpModel.setLpName("orgLpModel");

                        orgLpModel = LpSolve.readLp(LpFileName, LpSolve.NORMAL, orgLpModel.getLpName());

                        //for testing and debugging purposes
                        String orgOutputFile = "OrgLpModel_"+outputFileName;
                        System.out.println("orginal lp model is in file: "+ orgOutputFile );
                        orgLpModel.setOutputfile(orgOutputFile );
                        orgLpModel.printLp();

                }catch(LpSolveException lpe){System.out.println("LpSolveException in LagrangeConstructor: " +
lpe.getMessage());}

        }//end of overloaded constructor

        public void beginLagrangeRelaxations()
        {
                initMultipliers();                              //initialize lagrange multipliers to 0

                setupLR();                                                      //now setup a second lp model
with the lagrangean objective function f +Li(Ax-b)

                addConstraints();                               //add constraints not to be relaxed

                solveLR();

        }//end of beginLagrangeRelaxations

        public void initMultipliers()
        {
                lrMult = new double[orgLpModel.getNrows()+1];

                for(int i = 0; i < lrMult.length; i++)
                        lrMult[i] = 0;

        }//end of initMultipliers

        //creating relaxed LP model and adding limits and variable types
        public void setupLR()
        {
                double[] LR_objFnRow = new double[orgLpModel.getNcolumns()+1];

                int servLoc_Indi=0, throughputServLocConstr=0, networkConstr=0, delayServIndi=0, delayRounding=0,
delayLookup=0, delayCompute=0, delayMeet=0, safety=0;

                numConstrNotRelaxed = 0;
```

```
                    try
                    {
                            /////constraints to be rlaxed

//                          servLoc_Indi = pi.n+pi.n;
                                    //1,2
                            throughputServLocConstr = 3*(pi.n*pi.s);
                    //3,4,5

                            //count only those constraints where the edges are being used                  //6
                            int edgesUsed = 0;
                            boolean used;
                            for(int l=0; l < pi.e; l++)
                            {
                                    used = false;
                                    for(int m = 0; m < pi.p; m++)
                                    {
                                            if(pi.L[m][l] > 0)
                                                    used=true;
                                    }
                                    if(used==true)
                                            edgesUsed++;
                            }

                            networkConstr = edgesUsed;
                                    //6  -- 2 edges arenot used in the scenario

                            delayServIndi = (pi.p*pi.s)+(pi.p*pi.s);
                    //7,8
                            delayRounding = 2*edgesUsed + 2*pi.e;
                    //9,T9,10,T10

                            //num of constraints dependent on bandwidth
                    //11,T11,13,T13
                            int lookupIndexRange=0;
                            for(int l = 0; l < pi.e;l++)
                                    lookupIndexRange += (pi.b[l]/pi.qInterval)+1;

                            delayLookup = /*4*/3*(lookupIndexRange)+pi.e;
            //11,T11,13,T13 and 12

                            delayCompute = (2*pi.e)+(2*pi.p);
                            //14,T14,15,T15

                            //num of constraints depends on throughput requested and routing matrix
                            int servProvider = 0;
                            for(int i = 0; i < pi.n; i++)
                            //16,17,18
                                    for(int j = 0; j < pi.n; j++)
                                            for(int k = 0; k < pi.s; k++)
                                            {
                                                    if(pi.T[j][k] > 0)
                                                            for(int m = 0; m < pi.p; m++)
                                                                    if(pi.R[i][j][m] > 0)
                                                                            servProvider++;
                                            }

                            delayMeet = 3*servProvider;
                                    //16,17,18

                            safety = 2*(pi.n*pi.s);
                                            //19,20

                            numConstrNotRelaxed= servLoc_Indi + throughputServLocConstr + networkConstr +
delayServIndi + delayRounding + delayLookup + delayCompute + delayMeet + safety;


                            //////creating LP Model

                            lr_lps = LpSolve.makeLp(numConstrNotRelaxed,orgLpModel.getNcolumns());
                            lr_lps.setLpName("LR LpModel");

                            orgLpModel.getRow(0, LR_objFnRow);                                              //get obj
fn from org lp model

                            lr_lps.setObjFn(LR_objFnRow);                                                   //set it
as the obj fn for the lagrangean relaxation lp model

                            for(int i = 1; i <= orgLpModel.getNcolumns(); i++)
                            {
                                    lr_lps.setColName(i, orgLpModel.getColName(i));
            //set the column name for easy inferencing
                                    lr_lps.setBinary(i,true);
                            //and variable type
                            }
```

```
				for(int i = 0; i < pi.p; i++)
					for(int j = 0; j < pi.s; j++)
					{
						//find Zij in model and set it to an int
						String colName = "Z"+i+"_"+j;
						int index = lr_lps.getNameindex(colName, false);
						lr_lps.setInt(index, true);
						lr_lps.setUpbo(index, orgLpModel.getUpbo(index));
					}

				for(int l = 0; l < pi.e; l++)
				{
					//find Vl in model and set it to an int
					String colName = "V"+l;
					int index = lr_lps.getNameindex(colName, false);
					lr_lps.setInt(index, true);
					lr_lps.setUpbo(index, orgLpModel.getUpbo(index));

					//find al in model and set it to an int
					colName = "a"+l;
					index = lr_lps.getNameindex(colName, false);
					lr_lps.setInt(index, true);
					lr_lps.setUpbo(index, orgLpModel.getUpbo(index));

					//find Gl in model and set it to an int
					colName = "G"+l;
					index = lr_lps.getNameindex(colName, false);
					lr_lps.setInt(index, true);
					lr_lps.setUpbo(index, orgLpModel.getUpbo(index));
				}

				for(int m = 0; m < pi.p; m++)
				{
					//find Hm in model and set it to an int
					String colName = "H"+m;
					int index = lr_lps.getNameindex(colName, false);
					lr_lps.setInt(index, true);
					lr_lps.setUpbo(index, orgLpModel.getUpbo(index));
				}
			}catch(LpSolveException lpe){System.out.println("LpSolveException in setupLR: " +
lpe.getMessage());}

		}//end of setupLR


		public void addConstraints()
		{
			double[] LR_const = new double[orgLpModel.getNcolumns()+1];			//the constraints
not to be relaxed
			ignoreConstr = new int[orgLpModel.getNrows()+1];

			for(int i = 0; i < ignoreConstr.length; i++)
				ignoreConstr[i] = 0;

			try
			{
				int j = 1;

				////servLoc_Indi

/*				//do not relax constraint#1 -- throughput constraint
				for(int i = lpi.counters[0][0]; i < lpi.counters[1][0]; i++,j++)
				{
					//get org row from orgLpModel
					orgLpModel.getRow(i, LR_const);

					//keep track of constraints not to be considered in lagrange
					ignoreConstr[i] = 1;

					//add it to the lagrangean relaxation lp model
					double rhs = orgLpModel.getRh(i);
					lr_lps.setRow(j, LR_const);
					lr_lps.setRh(j, rhs);
					lr_lps.setConstrType(j,orgLpModel.getConstrType(i));
				}//end of adding constratint#1

				//do not relax constraint#2 -- throughput constraint
				for(int i = lpi.counters[1][0]; i < lpi.counters[2][0]; i++,j++)
				{
					//get org row from orgLpModel
					orgLpModel.getRow(i, LR_const);

					//keep track of constraints not to be considered in lagrange
```

```
          ignoreConstr[i] = 1;

          //add it to the lagrangean relaxation lp model
          double rhs = orgLpModel.getRh(i);
          lr_lps.setRow(j, LR_const);
          lr_lps.setRh(j, rhs);
          lr_lps.setConstrType(j,orgLpModel.getConstrType(i));
}//end of adding constratint#2
*/

////throughputServLocConstr

//do not relax constraint#3 -- throughput constraint
for(int i = lpi.counters[2][0]; i < lpi.counters[3][0]; i++,j++)
{
          //get org row from orgLpModel
          orgLpModel.getRow(i, LR_const);

          //keep track of constraints not to be considered in lagrange
          ignoreConstr[i] = 1;

          //add it to the lagrangean relaxation lp model
          double rhs = orgLpModel.getRh(i);
          lr_lps.setRow(j, LR_const);
          lr_lps.setRh(j, rhs);
          lr_lps.setConstrType(j,orgLpModel.getConstrType(i));
}//end of adding constratint#3

//do not relax constraint#4 -- throughput constraint
for(int i = lpi.counters[3][0]; i < lpi.counters[4][0]; i++,j++)
{
          //get org row from orgLpModel
          orgLpModel.getRow(i, LR_const);

          //keep track of constraints not to be considered in lagrange
          ignoreConstr[i] = 1;

          //add it to the lagrangean relaxation lp model
          double rhs = orgLpModel.getRh(i);
          lr_lps.setRow(j, LR_const);
          lr_lps.setRh(j, rhs);
          lr_lps.setConstrType(j,orgLpModel.getConstrType(i));
}//end of adding constratint#4

//do not relax constraint#5 -- throughput constraint
for(int i = lpi.counters[4][0]; i < lpi.counters[5][0]; i++,j++)
{
          //get org row from orgLpModel
          orgLpModel.getRow(i, LR_const);

          //keep track of constraints not to be considered in lagrange
          ignoreConstr[i] = 1;

          //add it to the lagrangean relaxation lp model
          double rhs = orgLpModel.getRh(i);
          lr_lps.setRow(j, LR_const);
          lr_lps.setRh(j, rhs);
          lr_lps.setConstrType(j,orgLpModel.getConstrType(i));
}//end of adding constratint#5


////networkConstr

//do not relax constraint#6 -- network
for(int i = lpi.counters[5][0]; i < lpi.counters[6][0]; i++,j++)
{
          //get org row from orgLpModel
          orgLpModel.getRow(i, LR_const);

          //keep track of constraints not to be considered in lagrange
          ignoreConstr[i] = 1;

          //add it to the lagrangean relaxation lp model
          double rhs = orgLpModel.getRh(i);
          lr_lps.setRow(j, LR_const);
          lr_lps.setRh(j, rhs);
          lr_lps.setConstrType(j,orgLpModel.getConstrType(i));
}//end of adding constratint#6


////delayServIndi

//do not relax constraint#7
for(int i = lpi.counters[6][0]; i < lpi.counters[7][0]; i++,j++)
{
```

```
        //get org row from orgLpModel
        orgLpModel.getRow(i, LR_const);

        //keep track of constraints not to be considered in lagrange
        ignoreConstr[i] = 1;

        //add it to the lagrangean relaxation lp model
        double rhs = orgLpModel.getRh(i);
        lr_lps.setRow(j, LR_const);
        lr_lps.setRh(j, rhs);
        lr_lps.setConstrType(j,orgLpModel.getConstrType(i));
}//end of adding constratint#7

//do not relax constraint#8
for(int i = lpi.counters[7][0]; i < lpi.counters[8][0]; i++,j++)
{
        //get org row from orgLpModel
        orgLpModel.getRow(i, LR_const);

        //keep track of constraints not to be considered in lagrange
        ignoreConstr[i] = 1;

        //add it to the lagrangean relaxation lp model
        double rhs = orgLpModel.getRh(i);
        lr_lps.setRow(j, LR_const);
        lr_lps.setRh(j, rhs);
        lr_lps.setConstrType(j,orgLpModel.getConstrType(i));
}//end of adding constratint#8

////delayRounding

//do not relax constraint#9
for(int i = lpi.counters[8][0]; i < lpi.counters[9][0]; i++,j++)
{
        //get org row from orgLpModel
        orgLpModel.getRow(i, LR_const);

        //keep track of constraints not to be considered in lagrange
        ignoreConstr[i] = 1;

        //add it to the lagrangean relaxation lp model
        double rhs = orgLpModel.getRh(i);
        lr_lps.setRow(j, LR_const);
        lr_lps.setRh(j, rhs);
        lr_lps.setConstrType(j,orgLpModel.getConstrType(i));
}//end of adding constratint#9

//do not relax constraint#T9
for(int i = lpi.counters[9][0]; i < lpi.counters[10][0]; i++,j++)
{
        //get org row from orgLpModel
        orgLpModel.getRow(i, LR_const);

        //keep track of constraints not to be considered in lagrange
        ignoreConstr[i] = 1;

        //add it to the lagrangean relaxation lp model
        double rhs = orgLpModel.getRh(i);
        lr_lps.setRow(j, LR_const);
        lr_lps.setRh(j, rhs);
        lr_lps.setConstrType(j,orgLpModel.getConstrType(i));
}//end of adding constratint#T9

//do not relax constraint#10
for(int i = lpi.counters[10][0]; i < lpi.counters[11][0]; i++,j++)
{
        //get org row from orgLpModel
        orgLpModel.getRow(i, LR_const);

        //keep track of constraints not to be considered in lagrange
        ignoreConstr[i] = 1;

        //add it to the lagrangean relaxation lp model
        double rhs = orgLpModel.getRh(i);
        lr_lps.setRow(j, LR_const);
        lr_lps.setRh(j, rhs);
        lr_lps.setConstrType(j,orgLpModel.getConstrType(i));
}//end of adding constratint#10

//do not relax constraint#T10
for(int i = lpi.counters[11][0]; i < lpi.counters[12][0]; i++,j++)
{
        //get org row from orgLpModel
        orgLpModel.getRow(i, LR_const);
```

```
        //keep track of constraints not to be considered in lagrange
        ignoreConstr[i] = 1;

        //add it to the lagrangean relaxation lp model
        double rhs = orgLpModel.getRh(i);
        lr_lps.setRow(j, LR_const);
        lr_lps.setRh(j, rhs);
        lr_lps.setConstrType(j,orgLpModel.getConstrType(i));
}//end of adding constratint#T10


////delayLookup

//do not relax constraint#11
for(int i = lpi.counters[12][0]; i < lpi.counters[13][0]; i++,j++)
{
        //get org row from orgLpModel
        orgLpModel.getRow(i, LR_const);

        //keep track of constraints not to be considered in lagrange
        ignoreConstr[i] = 1;

        //add it to the lagrangean relaxation lp model
        double rhs = orgLpModel.getRh(i);
        lr_lps.setRow(j, LR_const);
        lr_lps.setRh(j, rhs);
        lr_lps.setConstrType(j,orgLpModel.getConstrType(i));
}//end of adding constratint#11

//do not relax constraint#T11
for(int i = lpi.counters[13][0]; i < lpi.counters[14][0]; i++,j++)
{
        //get org row from orgLpModel
        orgLpModel.getRow(i, LR_const);

        //keep track of constraints not to be considered in lagrange
        ignoreConstr[i] = 1;

        //add it to the lagrangean relaxation lp model
        double rhs = orgLpModel.getRh(i);
        lr_lps.setRow(j, LR_const);
        lr_lps.setRh(j, rhs);
        lr_lps.setConstrType(j,orgLpModel.getConstrType(i));
}//end of adding constratint#T11

//do not relax constraint#12
for(int i = lpi.counters[14][0]; i < lpi.counters[15][0]; i++,j++)
{
        //get org row from orgLpModel
        orgLpModel.getRow(i, LR_const);

        //keep track of constraints not to be considered in lagrange
        ignoreConstr[i] = 1;

        //add it to the lagrangean relaxation lp model
        double rhs = orgLpModel.getRh(i);
        lr_lps.setRow(j, LR_const);
        lr_lps.setRh(j, rhs);
        lr_lps.setConstrType(j,orgLpModel.getConstrType(i));
}//end of adding constratint#12

/*
//do not relax constraint#13
for(int i = lpi.counters[15][0]; i < lpi.counters[16][0]; i++,j++)
{
        //get org row from orgLpModel
        orgLpModel.getRow(i, LR_const);

        //keep track of constraints not to be considered in lagrange
        ignoreConstr[i] = 1;

        //add it to the lagrangean relaxation lp model
        double rhs = orgLpModel.getRh(i);
        lr_lps.setRow(j, LR_const);
        lr_lps.setRh(j, rhs);
        lr_lps.setConstrType(j,orgLpModel.getConstrType(i));
}//end of adding constratint#13
*/
//do not relax constraint#T13
for(int i = lpi.counters[16][0]; i < lpi.counters[17][0]; i++,j++)
{
        //get org row from orgLpModel
        orgLpModel.getRow(i, LR_const);

        //keep track of constraints not to be considered in lagrange
        ignoreConstr[i] = 1;
```

```
        //add it to the lagrangean relaxation lp model
        double rhs = orgLpModel.getRh(i);
        lr_lps.setRow(j, LR_const);
        lr_lps.setRh(j, rhs);
        lr_lps.setConstrType(j,orgLpModel.getConstrType(i));
}//end of adding constratint#T13


////delayCompute

//do not relax constraint#14
for(int i = lpi.counters[17][0]; i < lpi.counters[18][0]; i++,j++)
{
        //get org row from orgLpModel
        orgLpModel.getRow(i, LR_const);

        //keep track of constraints not to be considered in lagrange
        ignoreConstr[i] = 1;

        //add it to the lagrangean relaxation lp model
        double rhs = orgLpModel.getRh(i);
        lr_lps.setRow(j, LR_const);
        lr_lps.setRh(j, rhs);
        lr_lps.setConstrType(j,orgLpModel.getConstrType(i));
}//end of adding constratint#14

//do not relax constraint#T14
for(int i = lpi.counters[18][0]; i < lpi.counters[19][0]; i++,j++)
{
        //get org row from orgLpModel
        orgLpModel.getRow(i, LR_const);

        //keep track of constraints not to be considered in lagrange
        ignoreConstr[i] = 1;

        //add it to the lagrangean relaxation lp model
        double rhs = orgLpModel.getRh(i);
        lr_lps.setRow(j, LR_const);
        lr_lps.setRh(j, rhs);
        lr_lps.setConstrType(j,orgLpModel.getConstrType(i));
}//end of adding constratint#T14

//do not relax constraint#15
for(int i = lpi.counters[19][0]; i < lpi.counters[20][0]; i++,j++)
{
        //get org row from orgLpModel
        orgLpModel.getRow(i, LR_const);

        //keep track of constraints not to be considered in lagrange
        ignoreConstr[i] = 1;

        //add it to the lagrangean relaxation lp model
        double rhs = orgLpModel.getRh(i);
        lr_lps.setRow(j, LR_const);
        lr_lps.setRh(j, rhs);
        lr_lps.setConstrType(j,orgLpModel.getConstrType(i));
}//end of adding constratint#15

//do not relax constraint#T15
for(int i = lpi.counters[20][0]; i < lpi.counters[21][0]; i++,j++)
{
        //get org row from orgLpModel
        orgLpModel.getRow(i, LR_const);

        //keep track of constraints not to be considered in lagrange
        ignoreConstr[i] = 1;

        //add it to the lagrangean relaxation lp model
        double rhs = orgLpModel.getRh(i);
        lr_lps.setRow(j, LR_const);
        lr_lps.setRh(j, rhs);
        lr_lps.setConstrType(j,orgLpModel.getConstrType(i));
}//end of adding constratint#T15


/////delayMeet

//do not relax constraint#16
for(int i = lpi.counters[21][0]; i < lpi.counters[22][0]; i++,j++)
{
        //get org row from orgLpModel
        orgLpModel.getRow(i, LR_const);

        //keep track of constraints not to be considered in lagrange
```

```
            ignoreConstr[i] = 1;

            //add it to the lagrangean relaxation lp model
            double rhs = orgLpModel.getRh(i);
            lr_lps.setRow(j, LR_const);
            lr_lps.setRh(j, rhs);
            lr_lps.setConstrType(j,orgLpModel.getConstrType(i));
    }//end of adding constratint#16

    //do not relax constraint#17
    for(int i = lpi.counters[22][0]; i < lpi.counters[23][0]; i++,j++)
    {
            //get org row from orgLpModel
            orgLpModel.getRow(i, LR_const);

            //keep track of constraints not to be considered in lagrange
            ignoreConstr[i] = 1;

            //add it to the lagrangean relaxation lp model
            double rhs = orgLpModel.getRh(i);
            lr_lps.setRow(j, LR_const);
            lr_lps.setRh(j, rhs);
            lr_lps.setConstrType(j,orgLpModel.getConstrType(i));
    }//end of adding constratint#17

    //do not relax constraint#18
    for(int i = lpi.counters[23][0]; i < lpi.counters[24][0]; i++,j++)
    {
            //get org row from orgLpModel
            orgLpModel.getRow(i, LR_const);

            //keep track of constraints not to be considered in lagrange
            ignoreConstr[i] = 1;

            //add it to the lagrangean relaxation lp model
            double rhs = orgLpModel.getRh(i);
            lr_lps.setRow(j, LR_const);
            lr_lps.setRh(j, rhs);
            lr_lps.setConstrType(j,orgLpModel.getConstrType(i));
    }//end of adding constratint#18


    /////safety

    //do not relax constraint#19
    for(int i = lpi.counters[24][0]; i < lpi.counters[25][0]; i++,j++)
    {
            //get org row from orgLpModel
            orgLpModel.getRow(i, LR_const);

            //keep track of constraints not to be considered in lagrange
            ignoreConstr[i] = 1;

            //add it to the lagrangean relaxation lp model
            double rhs = orgLpModel.getRh(i);
            lr_lps.setRow(j, LR_const);
            lr_lps.setRh(j, rhs);
            lr_lps.setConstrType(j,orgLpModel.getConstrType(i));
    }//end of adding constratint#19

    //do not relax constraint#20
    for(int i = lpi.counters[25][0]; i < lpi.counters[26][0]; i++,j++)
    {
            //get org row from orgLpModel
            orgLpModel.getRow(i, LR_const);

            //keep track of constraints not to be considered in lagrange
            ignoreConstr[i] = 1;

            //add it to the lagrangean relaxation lp model
            double rhs = orgLpModel.getRh(i);
            lr_lps.setRow(j, LR_const);
            lr_lps.setRh(j, rhs);
            lr_lps.setConstrType(j,orgLpModel.getConstrType(i));
    }//end of adding constratint#20


    //debugging
    lr_lps.setOutputfile("LR_Model_"+outputFileName);
    System.out.println("from method: addConstraints, printing LR model in file: " +
"LR_Model_"+outputFileName);
    lr_lps.printLp();

}//end of try
```

```
                catch(LpSolveException lpe){System.out.println("LpSolveException in method addConstraints: error is
: "+lpe.getMessage());}

        }//end of method addConstraints

        public void solveLR()
        {
                BufferedReader br = new BufferedReader(new InputStreamReader(System.in));
                String goon="";

                double[] LR_objFnRow = new double[orgLpModel.getNcolumns()+1];
        //the lagrangean obj fn
                double[] copyOfLrMult = new double[lrMult.length];
                double[] variables = new double[orgLpModel.getNcolumns()+1];
        //value of variables
                double[] copyOfVariables = new double[orgLpModel.getNcolumns()+1];                    //copy of
variables
                int[] nzIndex = new int[orgLpModel.getNcolumns()];
                //the variable indices

                double stepK = 2, bestRelaxedValue = 0, currentRelaxedValue = 0, constant = 0;

                double copyStepK=stepK, optimal=0, copyBestRelaxedValue = 0, copyCurrentRelaxedValue=0;
                int copyTotalIter = 0;
                boolean copyKTooSmall = false;;

                int m = 0, totalIter = 0;

                boolean kTooSmall = false, allConstrMet = true, constrMet = false;

                String msg="";
                int statusLR=0, statusSolve=0;

                //Get current time
                long start = System.currentTimeMillis();

                //set nzIndex the indices of the variables
                for(int i = 0; i < orgLpModel.getNcolumns(); i++)
                        nzIndex[i] = i+1;

                try{

                        do{
                                System.out.println("iteration = " + totalIter + " step size = " + stepK);

                                allConstrMet = true;

                                constant = createLRObjFn(LR_objFnRow);
                                lr_lps.setObjFn(LR_objFnRow);

                                //set timeout
                                lr_lps.setTimeout(120);
                                                //2 minutes = 120 sec to find a solution
                                lr_lps.setVerbose(LpSolve.MSG_NONE);
                                statusSolve = lr_lps.solve();
                                //step 2

                                if(statusSolve == 7)
                                {
                                        System.out.println("timed out, unable to find solution to lagrangean
function.");

                                        allConstrMet = false;
                                }
                                else
                                {
                                        //get variables values
                                        lr_lps.getVariables(variables);

                                        currentRelaxedValue = lr_lps.getObjective()+constant;

                                        System.out.println("constant = " + constant + " currentRelaxedValue =
" + currentRelaxedValue + " and bestRelaxedValue = " + bestRelaxedValue+"\n");

                                        //step 3 and 4
                                        //get each constraint value from orgLpModel and check for constraints
violated with this solution set
                                        for(int i = 1; i <= orgLpModel.getNrows(); i++)
                                        {
                                                if(ignoreConstr[i]==0)
                                                {
                                                        System.out.println("checking constraint = " + i);

                                                        double rhVal = orgLpModel.getConstrValue(i,
nzIndex.length, variables, nzIndex);
```

```
                        if(rhVal != orgLpModel.getRh(i))
                        {
                                if(rhVal < orgLpModel.getRh(i))       //positive slack, if rhval < constr.RH
                                        lrMult[i] -= stepK;           //decrease lambda
                                else if(rhVal > orgLpModel.getRh(i))  //constraint violated
                                {
        System.out.println("violated..." + rhVal + " > " + orgLpModel.getRh(i));
                                        lrMult[i] += stepK;           //increase lambda
                                        allConstrMet = false;
                                }
                        }//if values arent equal
                }//if constraint is relaxed
        }//validate all constraints in org lp model

        if(currentRelaxedValue > bestRelaxedValue)
        {
                bestRelaxedValue = currentRelaxedValue;
                m = 0;
        }
        else
                m++;

        if(allConstrMet == true)                                      //all
constraints have been met keep track of
        {                                 //optimal value and LpModel that gave those results
                if(constrMet == false)
                {
                        //copy variables array
                        for(int i = 0; i < orgLpModel.getNcolumns()+1;
i++)
                                copyOfVariables[i] = variables[i];

                        copyStepK = stepK;
                        copyTotalIter = totalIter;
                        copyBestRelaxedValue = bestRelaxedValue;
                        copyCurrentRelaxedValue = currentRelaxedValue;
                        copyKTooSmall = kTooSmall;
                        constrMet = true;
                        System.out.println("allConstrMet is true, copied
model");
                }//first time constraints met
                else
                {
                        System.out.println("all constranints met again,
see if the solution is better than what was already found");

                        if(bestRelaxedValue > copyBestRelaxedValue)
                        {
                                //copy variables array
                                for(int i = 0; i <
orgLpModel.getNcolumns()+1; i++)
                                        copyOfVariables[i] =
variables[i];

                                copyStepK = stepK;
                                copyTotalIter = totalIter;
                                copyBestRelaxedValue =
bestRelaxedValue;
                                copyCurrentRelaxedValue =
currentRelaxedValue;
                                copyKTooSmall = kTooSmall;
                                System.out.println("BETTER SOLUTION = "
+ bestRelaxedValue + " OLDER SOLN = " + copyBestRelaxedValue);
                                System.out.println("BETTER SOLUTION
FOUND and RECORDED");
                        }//if current model has a MORE optimal value than
bestModel then copy
                }//else --- next time constraints met only copy model if
obj fn value is less than model already copied
                allConstrMet = false;
        }//when all constr met

        if(m == 3)                                                    //step 5
        {
                stepK /= 2;
                m = 0;                                                //reset m
        }

        kTooSmall = checkStepSize(stepK);
```

```
                                        totalIter++;
                                }//end of else solver doesnot timeout

                        }while(!kTooSmall);

                        // Get elapsed time in milliseconds
                        long elapsedTimeMillis = System.currentTimeMillis()-start;

                        if(constrMet == true)
                        {
                                msg = "atleast once all constraints were met";
                                statusLR = 1;
                        }//allConstrMet
                        else if(statusSolve ==7)
                        {
                                msg = "solver timed out";
                                statusLR = 7;
                        }//solver timedout
                        else
                        {
                                msg = "step size too small";
                                statusLR = 0;
                        }//iterations too small

                        print(copyKTooSmall, copyTotalIter, copyStepK, copyCurrentRelaxedValue,
copyBestRelaxedValue, elapsedTimeMillis, msg, statusLR, copyOfVariables);


                }//end try
                catch(LpSolveException lpe){System.out.println("LpSolveException in solveLR: "+ lpe.getMessage());}
//              catch(IOException ioe){System.out.println("IOExceptions in solveLR: "+ ioe.getMessage());}

        }//end of solveLR


        //multiply the constraints to be relaxed with appropriate mutlipliers
        //and these constraints to the obj fn (i.e. collect all like variables together
        //and explicitly copy the lagrangean obj fn into objFnRow
        public double createLRObjFn(double[] objFnRow)
        {
                double[] rowFromLP = new double[orgLpModel.getNcolumns()+1];
                double[] rowForLR = new double[orgLpModel.getNcolumns()+1];
                double rhsColumn = 0;

                try
                {
                        orgLpModel.getRow(0, rowForLR);                                         //get obj
fn from org lp model

                        for(int i = 1; i <= orgLpModel.getNrows(); i++)
                        {
                                if(ignoreConstr[i] == 0)
                                {
                                        //get a row from the orgLP
                                        orgLpModel.getRow(i, rowFromLP);

                                        //computing L1(A1x-b1) + L2(A2x-b2) ...
                                        //L1*A1
                                        for(int j = 0; j < rowFromLP.length; j++)
                        //column 0 is just empty
                                                rowFromLP[j] = rowFromLP[j] * lrMult[i];

                                        //collect all constants -L1*b1*(-1)
                                        rhsColumn += (-1)*orgLpModel.getRh(i)*lrMult[i];

                                        //add the variables together, creating one obj fn with the relaxed
cosntraints
                                        for(int j = 0; j < rowForLR.length; j++)
                                                rowForLR[j] += rowFromLP[j];

                                }//if constraint is not to be ignored
                        }//end of for loop i

                        //explicilty copy the rowForLR into objFnRow
                        //rowForLR contains the objFn + lagConstraints
                        for(int i = 0; i < rowForLR.length; i++)
                                objFnRow[i] = rowForLR[i];

                }catch(LpSolveException lpe){System.out.println("LpSolveException in createLRObjFn: " +
lpe.getMessage());}

                return rhsColumn;

        }//end of createLRObjFnRow
```

```
	public boolean checkStepSize(double stepK)
	{
		if(Math.abs(stepK) < (0.05))
			return true;
		else
			return false;

	}//end of checkStepSize

	public void copyMultipliers(double c[])
	{
		for(int i = 0; i < c.length; i++)
			c[i] = lrMult[i];

	}//end of copyMultprs


	public void print(boolean kTooSmall, int totalIter, double stepK, double currentRelaxedValue, double
bestRelaxedValue, long elapsedTimeMillis, String msg, int status, double[] copyOfVariables)
	{
		try
		{
			String outputFile = "Variables__"+lrOutputFile;
			PrintWriter pw = new PrintWriter(new FileWriter(outputFileName));

			System.out.println("\n\n----------------------------------------------------------\n\n");
			System.out.println("Lagrangean relaxation output is in file: "+lrOutputFile);

			pw.println("---------------------------------------------------------------");
			pw.println("Quitting because: "+msg);
			pw.println("currentRelaxedValue = " + currentRelaxedValue + " best relaxed sol = "
+bestRelaxedValue);
			pw.println("step = " + stepK + "    total iterations = " + totalIter);
			pw.println("number of constraints not relaxed = " + numConstrNotRelaxed);
			pw.println("original model has " + orgLpModel.getNrows() + " constraints");
			pw.println("therefore we are relaxing " + (orgLpModel.getNrows()-numConstrNotRelaxed) + "
constraints");

			pw.println("\n\nElapsed Time: "+(float)elapsedTimeMillis/1000F);

			//print lagrange multipliers
			pw.println("\n\n The lagrange multipliers are: ");
			pw.println(printLagreMultipliers(lrMult));

			//use status to check which LpModel to print
			//print solutions
			switch(status)
			{
				case 0:				//no optimal solution found
					pw.println("Variables solutions are in file: "+outputFile);
					lr_lps.setOutputfile(outputFile);
					pw.println("objective function solution = " + lr_lps.getObjective());
					lr_lps.printSolution(2);
					break;
				case 1:
					pw.println("Variables solutions below...");
					for(int i = 0; i < orgLpModel.getNcolumns(); i++)
						pw.println(lr_lps.getColName(i+1) +
"\t\t"+copyOfVariables[i]);
					break;
				case 7:
					pw.println("Solver timed out. No solution found before time out...");
					break;
				default:
					System.out.println("in method print: unknown status, "+status);
			}//end of switch

			pw.println("---------------------------------------------------------------");
			System.out.println("------------------------------------------------------------");

			pw.close();

		}
		catch(IOException ioe){System.out.println("error in debugPrint: "+ioe.getMessage());}
		catch(LpSolveException lpe){System.out.println("LpSolveException in debugPrint:
"+lpe.getMessage());}

	}//end of print

	public String printLagreMultipliers(double l[])
	{
		String output = "";
```

```
            for(int i = 1; i < l.length; i++)
        //lambda0 is for row0 i.e. obj fn and not used
                    output += "lambda " + i + "  = " + l[i]+ "\n";

            return output;

        }//end of printLagreMultpliers

}//end of class LagrangeRelaxations
```

Appendix E

SLP Sample Scenario Input And Output Files

Sample SLP Problem Input File

```
n = 6
s = 3
p = 8
e = 8
L = 1;2,3;2;1,3;6;7,8;7;6,8
R = 1-2:1,2;1-3:3,4;6-4:5,6;6-5:7,8
b = 100,100,100,100,100,100,100,100
C =
910,100,80;120,100,150;100,100,100;100,120,120;100,100,110;100,110,90
T = 137,0,200;0,0,0;0,0,25;0,0,0;0,0,0;0,180,0
D = 1000,0,15;0,0,0;0,0,1000;0,0,0;0,0,0;0,1000,0
d = 20
```

Sample ILP File

```
min: 910 X0_0 + 100 X0_1 + 80 X0_2 + 120 X1_0 + 100 X1_1 + 150 X1_2 + 100 X2_0 + 100 X2_1 + 100 X2_2 + 100 X3_0 + 120
X3_1 + 120 X3_2 + 100 X4_0 + 100 X4_1 + 110 X4_2 + 100 X5_0 + 110 X5_1 + 90 X5_2 + 20 U0 + 20 U1 + 20 U2 + 20 U3 + 20
U4 + 20 U5;

/* service installation cost and location constraints */
/* constraint 1 */
X0_0 + X0_1 + X0_2  - 1000 U0 <= 0;
X1_0 + X1_1 + X1_2  - 1000 U1 <= 0;
X2_0 + X2_1 + X2_2  - 1000 U2 <= 0;
X3_0 + X3_1 + X3_2  - 1000 U3 <= 0;
X4_0 + X4_1 + X4_2  - 1000 U4 <= 0;
X5_0 + X5_1 + X5_2  - 1000 U5 <= 0;

/* constraint 2 */
 - X0_0  - X0_1  - X0_2  + U0 <= 0;
 - X1_0  - X1_1  - X1_2  + U1 <= 0;
 - X2_0  - X2_1  - X2_2  + U2 <= 0;
 - X3_0  - X3_1  - X3_2  + U3 <= 0;
 - X4_0  - X4_1  - X4_2  + U4 <= 0;
 - X5_0  - X5_1  - X5_2  + U5 <= 0;


/* throughput constraints */
/* constraint 3 */
 - Z16_0  - Z8_0  - Z9_0  - Z10_0  - Z11_0  <= -137;
 - Z16_1  - Z8_1  - Z9_1  - Z10_1  - Z11_1  <= -0;
 - Z16_2  - Z8_2  - Z9_2  - Z10_2  - Z11_2  <= -200;
 - Z0_0  - Z1_0  - Z17_0  <= -0;
 - Z0_1  - Z1_1  - Z17_1  <= -0;
 - Z0_2  - Z1_2  - Z17_2  <= -0;
 - Z2_0  - Z3_0  - Z18_0  <= -0;
 - Z2_1  - Z3_1  - Z18_1  <= -0;
 - Z2_2  - Z3_2  - Z18_2  <= -25;
 - Z19_0  - Z4_0  - Z5_0  <= -0;
 - Z19_1  - Z4_1  - Z5_1  <= -0;
 - Z19_2  - Z4_2  - Z5_2  <= -0;
 - Z20_0  - Z6_0  - Z7_0  <= -0;
 - Z20_1  - Z6_1  - Z7_1  <= -0;
 - Z20_2  - Z6_2  - Z7_2  <= -0;
 - Z12_0  - Z13_0  - Z14_0  - Z15_0  - Z21_0  <= -0;
 - Z12_1  - Z13_1  - Z14_1  - Z15_1  - Z21_1  <= -180;
 - Z12_2  - Z13_2  - Z14_2  - Z15_2  - Z21_2  <= -0;

/* constraint 4 */
X0_0 -  Z16_0 -  Z0_0 -  Z1_0 -  Z2_0 -  Z3_0 <= 0;
X0_1 -  Z16_1 -  Z0_1 -  Z1_1 -  Z2_1 -  Z3_1 <= 0;
X0_2 -  Z16_2 -  Z0_2 -  Z1_2 -  Z2_2 -  Z3_2 <= 0;
X1_0 -  Z8_0 -  Z9_0 -  Z17_0 <= 0;
X1_1 -  Z8_1 -  Z9_1 -  Z17_1 <= 0;
X1_2 -  Z8_2 -  Z9_2 -  Z17_2 <= 0;
X2_0 -  Z10_0 -  Z11_0 -  Z18_0 <= 0;
X2_1 -  Z10_1 -  Z11_1 -  Z18_1 <= 0;
X2_2 -  Z10_2 -  Z11_2 -  Z18_2 <= 0;
X3_0 -  Z19_0 -  Z12_0 -  Z13_0 <= 0;
X3_1 -  Z19_1 -  Z12_1 -  Z13_1 <= 0;
X3_2 -  Z19_2 -  Z12_2 -  Z13_2 <= 0;
X4_0 -  Z20_0 -  Z14_0 -  Z15_0 <= 0;
X4_1 -  Z20_1 -  Z14_1 -  Z15_1 <= 0;
X4_2 -  Z20_2 -  Z14_2 -  Z15_2 <= 0;
X5_0 -  Z4_0 -  Z5_0 -  Z6_0 -  Z7_0 -  Z21_0 <= 0;
X5_1 -  Z4_1 -  Z5_1 -  Z6_1 -  Z7_1 -  Z21_1 <= 0;
X5_2 -  Z4_2 -  Z5_2 -  Z6_2 -  Z7_2 -  Z21_2 <= 0;
/* constraint 5 */
 - 11000 X0_0 +   Z16_0 +   Z0_0 +   Z1_0 +   Z2_0 +   Z3_0 <= 0;
 - 11000 X0_1 +   Z16_1 +   Z0_1 +   Z1_1 +   Z2_1 +   Z3_1 <= 0;
 - 11000 X0_2 +   Z16_2 +   Z0_2 +   Z1_2 +   Z2_2 +   Z3_2 <= 0;
 - 11000 X1_0 +   Z8_0 +   Z9_0 +   Z17_0 <= 0;
 - 11000 X1_1 +   Z8_1 +   Z9_1 +   Z17_1 <= 0;
 - 11000 X1_2 +   Z8_2 +   Z9_2 +   Z17_2 <= 0;
 - 11000 X2_0 +   Z10_0 +   Z11_0 +   Z18_0 <= 0;
 - 11000 X2_1 +   Z10_1 +   Z11_1 +   Z18_1 <= 0;
 - 11000 X2_2 +   Z10_2 +   Z11_2 +   Z18_2 <= 0;
 - 11000 X3_0 +   Z19_0 +   Z12_0 +   Z13_0 <= 0;
 - 11000 X3_1 +   Z19_1 +   Z12_1 +   Z13_1 <= 0;
 - 11000 X3_2 +   Z19_2 +   Z12_2 +   Z13_2 <= 0;
 - 11000 X4_0 +   Z20_0 +   Z14_0 +   Z15_0 <= 0;
```

```
 - 11000 X4_1 +   Z20_1 +   Z14_1 +   Z15_1 <= 0;
 - 11000 X4_2 +   Z20_2 +   Z14_2 +   Z15_2 <= 0;
 - 11000 X5_0 +   Z4_0 +   Z5_0 +   Z6_0 +   Z7_0 +   Z21_0 <= 0;
 - 11000 X5_1 +   Z4_1 +   Z5_1 +   Z6_1 +   Z7_1 +   Z21_1 <= 0;
 - 11000 X5_2 +   Z4_2 +   Z5_2 +   Z6_2 +   Z7_2 +   Z21_2 <= 0;


/* network link capacity constraint */
/* constraint 6 */
Z0_0 + Z0_1 + Z0_2 + Z3_0 + Z3_1 + Z3_2 + Z8_0 + Z8_1 + Z8_2 + Z11_0 + Z11_1 + Z11_2 <= 100;
Z1_0 + Z1_1 + Z1_2 + Z2_0 + Z2_1 + Z2_2 + Z9_0 + Z9_1 + Z9_2 + Z10_0 + Z10_1 + Z10_2 <= 100;
Z1_0 + Z1_1 + Z1_2 + Z3_0 + Z3_1 + Z3_2 + Z9_0 + Z9_1 + Z9_2 + Z11_0 + Z11_1 + Z11_2 <= 100;


Z4_0 + Z4_1 + Z4_2 + Z7_0 + Z7_1 + Z7_2 + Z12_0 + Z12_1 + Z12_2 + Z15_0 + Z15_1 + Z15_2 <= 100;
Z5_0 + Z5_1 + Z5_2 + Z6_0 + Z6_1 + Z6_2 + Z13_0 + Z13_1 + Z13_2 + Z14_0 + Z14_1 + Z14_2 <= 100;
Z5_0 + Z5_1 + Z5_2 + Z7_0 + Z7_1 + Z7_2 + Z13_0 + Z13_1 + Z13_2 + Z15_0 + Z15_1 + Z15_2 <= 100;
Z16_0 + Z16_1 + Z16_2 <= 1000;
Z17_0 + Z17_1 + Z17_2 <= 1000;
Z18_0 + Z18_1 + Z18_2 <= 1000;
Z19_0 + Z19_1 + Z19_2 <= 1000;
Z20_0 + Z20_1 + Z20_2 <= 1000;
Z21_0 + Z21_1 + Z21_2 <= 1000;


/*********************** DELAY CONSTRAINTS ***********************/



/* path serivce indicator */
/* constraint 7 */
 - 11000 Y0_0 + Z0_0 <= 0;
 - 11000 Y0_1 + Z0_1 <= 0;
 - 11000 Y0_2 + Z0_2 <= 0;
 - 11000 Y1_0 + Z1_0 <= 0;
 - 11000 Y1_1 + Z1_1 <= 0;
 - 11000 Y1_2 + Z1_2 <= 0;
 - 11000 Y2_0 + Z2_0 <= 0;
 - 11000 Y2_1 + Z2_1 <= 0;
 - 11000 Y2_2 + Z2_2 <= 0;
 - 11000 Y3_0 + Z3_0 <= 0;
 - 11000 Y3_1 + Z3_1 <= 0;
 - 11000 Y3_2 + Z3_2 <= 0;
 - 11000 Y4_0 + Z4_0 <= 0;
 - 11000 Y4_1 + Z4_1 <= 0;
 - 11000 Y4_2 + Z4_2 <= 0;
 - 11000 Y5_0 + Z5_0 <= 0;
 - 11000 Y5_1 + Z5_1 <= 0;
 - 11000 Y5_2 + Z5_2 <= 0;
 - 11000 Y6_0 + Z6_0 <= 0;
 - 11000 Y6_1 + Z6_1 <= 0;
 - 11000 Y6_2 + Z6_2 <= 0;
 - 11000 Y7_0 + Z7_0 <= 0;
 - 11000 Y7_1 + Z7_1 <= 0;
 - 11000 Y7_2 + Z7_2 <= 0;
 - 11000 Y8_0 + Z8_0 <= 0;
 - 11000 Y8_1 + Z8_1 <= 0;
 - 11000 Y8_2 + Z8_2 <= 0;
 - 11000 Y9_0 + Z9_0 <= 0;
 - 11000 Y9_1 + Z9_1 <= 0;
 - 11000 Y9_2 + Z9_2 <= 0;
 - 11000 Y10_0 + Z10_0 <= 0;
 - 11000 Y10_1 + Z10_1 <= 0;
 - 11000 Y10_2 + Z10_2 <= 0;
 - 11000 Y11_0 + Z11_0 <= 0;
 - 11000 Y11_1 + Z11_1 <= 0;
 - 11000 Y11_2 + Z11_2 <= 0;
 - 11000 Y12_0 + Z12_0 <= 0;
 - 11000 Y12_1 + Z12_1 <= 0;
 - 11000 Y12_2 + Z12_2 <= 0;
 - 11000 Y13_0 + Z13_0 <= 0;
 - 11000 Y13_1 + Z13_1 <= 0;
 - 11000 Y13_2 + Z13_2 <= 0;
 - 11000 Y14_0 + Z14_0 <= 0;
 - 11000 Y14_1 + Z14_1 <= 0;
 - 11000 Y14_2 + Z14_2 <= 0;
 - 11000 Y15_0 + Z15_0 <= 0;
 - 11000 Y15_1 + Z15_1 <= 0;
 - 11000 Y15_2 + Z15_2 <= 0;
 - 11000 Y16_0 + Z16_0 <= 0;
 - 11000 Y16_1 + Z16_1 <= 0;
 - 11000 Y16_2 + Z16_2 <= 0;
```

```
- 11000 Y17_0 + Z17_0 <= 0;
- 11000 Y17_1 + Z17_1 <= 0;
- 11000 Y17_2 + Z17_2 <= 0;
- 11000 Y18_0 + Z18_0 <= 0;
- 11000 Y18_1 + Z18_1 <= 0;
- 11000 Y18_2 + Z18_2 <= 0;
- 11000 Y19_0 + Z19_0 <= 0;
- 11000 Y19_1 + Z19_1 <= 0;
- 11000 Y19_2 + Z19_2 <= 0;
- 11000 Y20_0 + Z20_0 <= 0;
- 11000 Y20_1 + Z20_1 <= 0;
- 11000 Y20_2 + Z20_2 <= 0;
- 11000 Y21_0 + Z21_0 <= 0;
- 11000 Y21_1 + Z21_1 <= 0;
- 11000 Y21_2 + Z21_2 <= 0;

/* constraint 8 */
 Y0_0 -  Z0_0 <= 0;
 Y0_1 -  Z0_1 <= 0;
 Y0_2 -  Z0_2 <= 0;
 Y1_0 -  Z1_0 <= 0;
 Y1_1 -  Z1_1 <= 0;
 Y1_2 -  Z1_2 <= 0;
 Y2_0 -  Z2_0 <= 0;
 Y2_1 -  Z2_1 <= 0;
 Y2_2 -  Z2_2 <= 0;
 Y3_0 -  Z3_0 <= 0;
 Y3_1 -  Z3_1 <= 0;
 Y3_2 -  Z3_2 <= 0;
 Y4_0 -  Z4_0 <= 0;
 Y4_1 -  Z4_1 <= 0;
 Y4_2 -  Z4_2 <= 0;
 Y5_0 -  Z5_0 <= 0;
 Y5_1 -  Z5_1 <= 0;
 Y5_2 -  Z5_2 <= 0;
 Y6_0 -  Z6_0 <= 0;
 Y6_1 -  Z6_1 <= 0;
 Y6_2 -  Z6_2 <= 0;
 Y7_0 -  Z7_0 <= 0;
 Y7_1 -  Z7_1 <= 0;
 Y7_2 -  Z7_2 <= 0;
 Y8_0 -  Z8_0 <= 0;
 Y8_1 -  Z8_1 <= 0;
 Y8_2 -  Z8_2 <= 0;
 Y9_0 -  Z9_0 <= 0;
 Y9_1 -  Z9_1 <= 0;
 Y9_2 -  Z9_2 <= 0;
 Y10_0 -  Z10_0 <= 0;
 Y10_1 -  Z10_1 <= 0;
 Y10_2 -  Z10_2 <= 0;
 Y11_0 -  Z11_0 <= 0;
 Y11_1 -  Z11_1 <= 0;
 Y11_2 -  Z11_2 <= 0;
 Y12_0 -  Z12_0 <= 0;
 Y12_1 -  Z12_1 <= 0;
 Y12_2 -  Z12_2 <= 0;
 Y13_0 -  Z13_0 <= 0;
 Y13_1 -  Z13_1 <= 0;
 Y13_2 -  Z13_2 <= 0;
 Y14_0 -  Z14_0 <= 0;
 Y14_1 -  Z14_1 <= 0;
 Y14_2 -  Z14_2 <= 0;
 Y15_0 -  Z15_0 <= 0;
 Y15_1 -  Z15_1 <= 0;
 Y15_2 -  Z15_2 <= 0;
 Y16_0 -  Z16_0 <= 0;
 Y16_1 -  Z16_1 <= 0;
 Y16_2 -  Z16_2 <= 0;
 Y17_0 -  Z17_0 <= 0;
 Y17_1 -  Z17_1 <= 0;
 Y17_2 -  Z17_2 <= 0;
 Y18_0 -  Z18_0 <= 0;
 Y18_1 -  Z18_1 <= 0;
 Y18_2 -  Z18_2 <= 0;
 Y19_0 -  Z19_0 <= 0;
 Y19_1 -  Z19_1 <= 0;
 Y19_2 -  Z19_2 <= 0;
 Y20_0 -  Z20_0 <= 0;
 Y20_1 -  Z20_1 <= 0;
 Y20_2 -  Z20_2 <= 0;
 Y21_0 -  Z21_0 <= 0;
 Y21_1 -  Z21_1 <= 0;
 Y21_2 -  Z21_2 <= 0;
```

```
/* compute load on link and round it up if necessary */
/* constraint 9 */
V0  - Z0_0 - Z0_1 - Z0_2 - Z3_0 - Z3_1 - Z3_2 - Z8_0 - Z8_1 - Z8_2 - Z11_0 - Z11_1 - Z11_2<= 0;
V1  - Z1_0 - Z1_1 - Z1_2 - Z2_0 - Z2_1 - Z2_2 - Z9_0 - Z9_1 - Z9_2 - Z10_0 - Z10_1 - Z10_2<= 0;
V2  - Z1_0 - Z1_1 - Z1_2 - Z3_0 - Z3_1 - Z3_2 - Z9_0 - Z9_1 - Z9_2 - Z11_0 - Z11_1 - Z11_2<= 0;
V3 <= 0;
V4 <= 0;
V5  - Z4_0 - Z4_1 - Z4_2 - Z7_0 - Z7_1 - Z7_2 - Z12_0 - Z12_1 - Z12_2 - Z15_0 - Z15_1 - Z15_2<= 0;
V6  - Z5_0 - Z5_1 - Z5_2 - Z6_0 - Z6_1 - Z6_2 - Z13_0 - Z13_1 - Z13_2 - Z14_0 - Z14_1 - Z14_2<= 0;
V7  - Z5_0 - Z5_1 - Z5_2 - Z7_0 - Z7_1 - Z7_2 - Z13_0 - Z13_1 - Z13_2 - Z15_0 - Z15_1 - Z15_2<= 0;
V8  - Z16_0 - Z16_1 - Z16_2<= 0;
V9  - Z17_0 - Z17_1 - Z17_2<= 0;
V10  - Z18_0 - Z18_1 - Z18_2<= 0;
V11  - Z19_0 - Z19_1 - Z19_2<= 0;
V12  - Z20_0 - Z20_1 - Z20_2<= 0;
V13  - Z21_0 - Z21_1 - Z21_2<= 0;
/* constraint T9 */
 - V0 + Z0_0 + Z0_1 + Z0_2 + Z3_0 + Z3_1 + Z3_2 + Z8_0 + Z8_1 + Z8_2 + Z11_0 + Z11_1 + Z11_2<= 0;
 - V1 + Z1_0 + Z1_1 + Z1_2 + Z2_0 + Z2_1 + Z2_2 + Z9_0 + Z9_1 + Z9_2 + Z10_0 + Z10_1 + Z10_2<= 0;
 - V2 + Z1_0 + Z1_1 + Z1_2 + Z3_0 + Z3_1 + Z3_2 + Z9_0 + Z9_1 + Z9_2 + Z11_0 + Z11_1 + Z11_2<= 0;
 - V3<= 0;
 - V4<= 0;
 - V5 + Z4_0 + Z4_1 + Z4_2 + Z7_0 + Z7_1 + Z7_2 + Z12_0 + Z12_1 + Z12_2 + Z15_0 + Z15_1 + Z15_2<= 0;
 - V6 + Z5_0 + Z5_1 + Z5_2 + Z6_0 + Z6_1 + Z6_2 + Z13_0 + Z13_1 + Z13_2 + Z14_0 + Z14_1 + Z14_2<= 0;
 - V7 + Z5_0 + Z5_1 + Z5_2 + Z7_0 + Z7_1 + Z7_2 + Z13_0 + Z13_1 + Z13_2 + Z15_0 + Z15_1 + Z15_2<= 0;
 - V8 + Z16_0 + Z16_1 + Z16_2<= 0;
 - V9 + Z17_0 + Z17_1 + Z17_2<= 0;
 - V10 + Z18_0 + Z18_1 + Z18_2<= 0;
 - V11 + Z19_0 + Z19_1 + Z19_2<= 0;
 - V12 + Z20_0 + Z20_1 + Z20_2<= 0;
 - V13 + Z21_0 + Z21_1 + Z21_2<= 0;

/* constraint 10 */
V0 - 50 a0 <= 0;
V1 - 50 a1 <= 0;
V2 - 50 a2 <= 0;
V3 - 50 a3 <= 0;
V4 - 50 a4 <= 0;
V5 - 50 a5 <= 0;
V6 - 50 a6 <= 0;
V7 - 50 a7 <= 0;
V8 - 50 a8 <= 0;
V9 - 50 a9 <= 0;
V10 - 50 a10 <= 0;
V11 - 50 a11 <= 0;
V12 - 50 a12 <= 0;
V13 - 50 a13 <= 0;

/* constraint T10 */
 - V0 + 50 a0 <= 0;
 - V1 + 50 a1 <= 0;
 - V2 + 50 a2 <= 0;
 - V3 + 50 a3 <= 0;
 - V4 + 50 a4 <= 0;
 - V5 + 50 a5 <= 0;
 - V6 + 50 a6 <= 0;
 - V7 + 50 a7 <= 0;
 - V8 + 50 a8 <= 0;
 - V9 + 50 a9 <= 0;
 - V10 + 50 a10 <= 0;
 - V11 + 50 a11 <= 0;
 - V12 + 50 a12 <= 0;
 - V13 + 50 a13 <= 0;

/* compute the index that matches the load on a link to the delay in the lookup table */

/* constraint 11 */
V0 - 0 - 1500 c0_0 <= 0;
V0 - 50 - 1500 c0_1 <= 0;
V0 - 100 - 1500 c0_2 <= 0;
V1 - 0 - 1500 c1_0 <= 0;
V1 - 50 - 1500 c1_1 <= 0;
V1 - 100 - 1500 c1_2 <= 0;
V2 - 0 - 1500 c2_0 <= 0;
V2 - 50 - 1500 c2_1 <= 0;
V2 - 100 - 1500 c2_2 <= 0;
V3 - 0 - 1500 c3_0 <= 0;
V3 - 50 - 1500 c3_1 <= 0;
V3 - 100 - 1500 c3_2 <= 0;
V4 - 0 - 1500 c4_0 <= 0;
V4 - 50 - 1500 c4_1 <= 0;
V4 - 100 - 1500 c4_2 <= 0;
V5 - 0 - 1500 c5_0 <= 0;
V5 - 50 - 1500 c5_1 <= 0;
V5 - 100 - 1500 c5_2 <= 0;
```

```
V6 - 0 - 1500 c6_0 <= 0;
V6 - 50 - 1500 c6_1 <= 0;
V6 - 100 - 1500 c6_2 <= 0;
V7 - 0 - 1500 c7_0 <= 0;
V7 - 50 - 1500 c7_1 <= 0;
V7 - 100 - 1500 c7_2 <= 0;
V8 - 0 - 1500 c8_0 <= 0;
V8 - 50 - 1500 c8_1 <= 0;
V8 - 100 - 1500 c8_2 <= 0;
V8 - 150 - 1500 c8_3 <= 0;
V8 - 200 - 1500 c8_4 <= 0;
V8 - 250 - 1500 c8_5 <= 0;
V8 - 300 - 1500 c8_6 <= 0;
V8 - 350 - 1500 c8_7 <= 0;
V8 - 400 - 1500 c8_8 <= 0;
V8 - 450 - 1500 c8_9 <= 0;
V8 - 500 - 1500 c8_10 <= 0;
V8 - 550 - 1500 c8_11 <= 0;
V8 - 600 - 1500 c8_12 <= 0;
V8 - 650 - 1500 c8_13 <= 0;
V8 - 700 - 1500 c8_14 <= 0;
V8 - 750 - 1500 c8_15 <= 0;
V8 - 800 - 1500 c8_16 <= 0;
V8 - 850 - 1500 c8_17 <= 0;
V8 - 900 - 1500 c8_18 <= 0;
V8 - 950 - 1500 c8_19 <= 0;
V8 - 1000 - 1500 c8_20 <= 0;
V9 - 0 - 1500 c9_0 <= 0;
V9 - 50 - 1500 c9_1 <= 0;
V9 - 100 - 1500 c9_2 <= 0;
V9 - 150 - 1500 c9_3 <= 0;
V9 - 200 - 1500 c9_4 <= 0;
V9 - 250 - 1500 c9_5 <= 0;
V9 - 300 - 1500 c9_6 <= 0;
V9 - 350 - 1500 c9_7 <= 0;
V9 - 400 - 1500 c9_8 <= 0;
V9 - 450 - 1500 c9_9 <= 0;
V9 - 500 - 1500 c9_10 <= 0;
V9 - 550 - 1500 c9_11 <= 0;
V9 - 600 - 1500 c9_12 <= 0;
V9 - 650 - 1500 c9_13 <= 0;
V9 - 700 - 1500 c9_14 <= 0;
V9 - 750 - 1500 c9_15 <= 0;
V9 - 800 - 1500 c9_16 <= 0;
V9 - 850 - 1500 c9_17 <= 0;
V9 - 900 - 1500 c9_18 <= 0;
V9 - 950 - 1500 c9_19 <= 0;
V9 - 1000 - 1500 c9_20 <= 0;
V10 - 0 - 1500 c10_0 <= 0;
V10 - 50 - 1500 c10_1 <= 0;
V10 - 100 - 1500 c10_2 <= 0;
V10 - 150 - 1500 c10_3 <= 0;
V10 - 200 - 1500 c10_4 <= 0;
V10 - 250 - 1500 c10_5 <= 0;
V10 - 300 - 1500 c10_6 <= 0;
V10 - 350 - 1500 c10_7 <= 0;
V10 - 400 - 1500 c10_8 <= 0;
V10 - 450 - 1500 c10_9 <= 0;
V10 - 500 - 1500 c10_10 <= 0;
V10 - 550 - 1500 c10_11 <= 0;
V10 - 600 - 1500 c10_12 <= 0;
V10 - 650 - 1500 c10_13 <= 0;
V10 - 700 - 1500 c10_14 <= 0;
V10 - 750 - 1500 c10_15 <= 0;
V10 - 800 - 1500 c10_16 <= 0;
V10 - 850 - 1500 c10_17 <= 0;
V10 - 900 - 1500 c10_18 <= 0;
V10 - 950 - 1500 c10_19 <= 0;
V10 - 1000 - 1500 c10_20 <= 0;
V11 - 0 - 1500 c11_0 <= 0;
V11 - 50 - 1500 c11_1 <= 0;
V11 - 100 - 1500 c11_2 <= 0;
V11 - 150 - 1500 c11_3 <= 0;
V11 - 200 - 1500 c11_4 <= 0;
V11 - 250 - 1500 c11_5 <= 0;
V11 - 300 - 1500 c11_6 <= 0;
V11 - 350 - 1500 c11_7 <= 0;
V11 - 400 - 1500 c11_8 <= 0;
V11 - 450 - 1500 c11_9 <= 0;
V11 - 500 - 1500 c11_10 <= 0;
V11 - 550 - 1500 c11_11 <= 0;
V11 - 600 - 1500 c11_12 <= 0;
V11 - 650 - 1500 c11_13 <= 0;
V11 - 700 - 1500 c11_14 <= 0;
V11 - 750 - 1500 c11_15 <= 0;
```

```
V11 - 800 - 1500 c11_16 <= 0;
V11 - 850 - 1500 c11_17 <= 0;
V11 - 900 - 1500 c11_18 <= 0;
V11 - 950 - 1500 c11_19 <= 0;
V11 - 1000 - 1500 c11_20 <= 0;
V12 - 0 - 1500 c12_0 <= 0;
V12 - 50 - 1500 c12_1 <= 0;
V12 - 100 - 1500 c12_2 <= 0;
V12 - 150 - 1500 c12_3 <= 0;
V12 - 200 - 1500 c12_4 <= 0;
V12 - 250 - 1500 c12_5 <= 0;
V12 - 300 - 1500 c12_6 <= 0;
V12 - 350 - 1500 c12_7 <= 0;
V12 - 400 - 1500 c12_8 <= 0;
V12 - 450 - 1500 c12_9 <= 0;
V12 - 500 - 1500 c12_10 <= 0;
V12 - 550 - 1500 c12_11 <= 0;
V12 - 600 - 1500 c12_12 <= 0;
V12 - 650 - 1500 c12_13 <= 0;
V12 - 700 - 1500 c12_14 <= 0;
V12 - 750 - 1500 c12_15 <= 0;
V12 - 800 - 1500 c12_16 <= 0;
V12 - 850 - 1500 c12_17 <= 0;
V12 - 900 - 1500 c12_18 <= 0;
V12 - 950 - 1500 c12_19 <= 0;
V12 - 1000 - 1500 c12_20 <= 0;
V13 - 0 - 1500 c13_0 <= 0;
V13 - 50 - 1500 c13_1 <= 0;
V13 - 100 - 1500 c13_2 <= 0;
V13 - 150 - 1500 c13_3 <= 0;
V13 - 200 - 1500 c13_4 <= 0;
V13 - 250 - 1500 c13_5 <= 0;
V13 - 300 - 1500 c13_6 <= 0;
V13 - 350 - 1500 c13_7 <= 0;
V13 - 400 - 1500 c13_8 <= 0;
V13 - 450 - 1500 c13_9 <= 0;
V13 - 500 - 1500 c13_10 <= 0;
V13 - 550 - 1500 c13_11 <= 0;
V13 - 600 - 1500 c13_12 <= 0;
V13 - 650 - 1500 c13_13 <= 0;
V13 - 700 - 1500 c13_14 <= 0;
V13 - 750 - 1500 c13_15 <= 0;
V13 - 800 - 1500 c13_16 <= 0;
V13 - 850 - 1500 c13_17 <= 0;
V13 - 900 - 1500 c13_18 <= 0;
V13 - 950 - 1500 c13_19 <= 0;
V13 - 1000 - 1500 c13_20 <= 0;
/* constraint T11 */
 - V0 + 0 - 1500 c0_0 <= 0;
 - V0 + 50 - 1500 c0_1 <= 0;
 - V0 + 100 - 1500 c0_2 <= 0;
 - V1 + 0 - 1500 c1_0 <= 0;
 - V1 + 50 - 1500 c1_1 <= 0;
 - V1 + 100 - 1500 c1_2 <= 0;
 - V2 + 0 - 1500 c2_0 <= 0;
 - V2 + 50 - 1500 c2_1 <= 0;
 - V2 + 100 - 1500 c2_2 <= 0;
 - V3 + 0 - 1500 c3_0 <= 0;
 - V3 + 50 - 1500 c3_1 <= 0;
 - V3 + 100 - 1500 c3_2 <= 0;
 - V4 + 0 - 1500 c4_0 <= 0;
 - V4 + 50 - 1500 c4_1 <= 0;
 - V4 + 100 - 1500 c4_2 <= 0;
 - V5 + 0 - 1500 c5_0 <= 0;
 - V5 + 50 - 1500 c5_1 <= 0;
 - V5 + 100 - 1500 c5_2 <= 0;
 - V6 + 0 - 1500 c6_0 <= 0;
 - V6 + 50 - 1500 c6_1 <= 0;
 - V6 + 100 - 1500 c6_2 <= 0;
 - V7 + 0 - 1500 c7_0 <= 0;
 - V7 + 50 - 1500 c7_1 <= 0;
 - V7 + 100 - 1500 c7_2 <= 0;
 - V8 + 0 - 1500 c8_0 <= 0;
 - V8 + 50 - 1500 c8_1 <= 0;
 - V8 + 100 - 1500 c8_2 <= 0;
 - V8 + 150 - 1500 c8_3 <= 0;
 - V8 + 200 - 1500 c8_4 <= 0;
 - V8 + 250 - 1500 c8_5 <= 0;
 - V8 + 300 - 1500 c8_6 <= 0;
 - V8 + 350 - 1500 c8_7 <= 0;
 - V8 + 400 - 1500 c8_8 <= 0;
 - V8 + 450 - 1500 c8_9 <= 0;
 - V8 + 500 - 1500 c8_10 <= 0;
 - V8 + 550 - 1500 c8_11 <= 0;
 - V8 + 600 - 1500 c8_12 <= 0;
```

```
- V8 + 650 - 1500 c8_13 <= 0;
- V8 + 700 - 1500 c8_14 <= 0;
- V8 + 750 - 1500 c8_15 <= 0;
- V8 + 800 - 1500 c8_16 <= 0;
- V8 + 850 - 1500 c8_17 <= 0;
- V8 + 900 - 1500 c8_18 <= 0;
- V8 + 950 - 1500 c8_19 <= 0;
- V8 + 1000 - 1500 c8_20 <= 0;
- V9 + 0 - 1500 c9_0 <= 0;
- V9 + 50 - 1500 c9_1 <= 0;
- V9 + 100 - 1500 c9_2 <= 0;
- V9 + 150 - 1500 c9_3 <= 0;
- V9 + 200 - 1500 c9_4 <= 0;
- V9 + 250 - 1500 c9_5 <= 0;
- V9 + 300 - 1500 c9_6 <= 0;
- V9 + 350 - 1500 c9_7 <= 0;
- V9 + 400 - 1500 c9_8 <= 0;
- V9 + 450 - 1500 c9_9 <= 0;
- V9 + 500 - 1500 c9_10 <= 0;
- V9 + 550 - 1500 c9_11 <= 0;
- V9 + 600 - 1500 c9_12 <= 0;
- V9 + 650 - 1500 c9_13 <= 0;
- V9 + 700 - 1500 c9_14 <= 0;
- V9 + 750 - 1500 c9_15 <= 0;
- V9 + 800 - 1500 c9_16 <= 0;
- V9 + 850 - 1500 c9_17 <= 0;
- V9 + 900 - 1500 c9_18 <= 0;
- V9 + 950 - 1500 c9_19 <= 0;
- V9 + 1000 - 1500 c9_20 <= 0;
- V10 + 0 - 1500 c10_0 <= 0;
- V10 + 50 - 1500 c10_1 <= 0;
- V10 + 100 - 1500 c10_2 <= 0;
- V10 + 150 - 1500 c10_3 <= 0;
- V10 + 200 - 1500 c10_4 <= 0;
- V10 + 250 - 1500 c10_5 <= 0;
- V10 + 300 - 1500 c10_6 <= 0;
- V10 + 350 - 1500 c10_7 <= 0;
- V10 + 400 - 1500 c10_8 <= 0;
- V10 + 450 - 1500 c10_9 <= 0;
- V10 + 500 - 1500 c10_10 <= 0;
- V10 + 550 - 1500 c10_11 <= 0;
- V10 + 600 - 1500 c10_12 <= 0;
- V10 + 650 - 1500 c10_13 <= 0;
- V10 + 700 - 1500 c10_14 <= 0;
- V10 + 750 - 1500 c10_15 <= 0;
- V10 + 800 - 1500 c10_16 <= 0;
- V10 + 850 - 1500 c10_17 <= 0;
- V10 + 900 - 1500 c10_18 <= 0;
- V10 + 950 - 1500 c10_19 <= 0;
- V10 + 1000 - 1500 c10_20 <= 0;
- V11 + 0 - 1500 c11_0 <= 0;
- V11 + 50 - 1500 c11_1 <= 0;
- V11 + 100 - 1500 c11_2 <= 0;
- V11 + 150 - 1500 c11_3 <= 0;
- V11 + 200 - 1500 c11_4 <= 0;
- V11 + 250 - 1500 c11_5 <= 0;
- V11 + 300 - 1500 c11_6 <= 0;
- V11 + 350 - 1500 c11_7 <= 0;
- V11 + 400 - 1500 c11_8 <= 0;
- V11 + 450 - 1500 c11_9 <= 0;
- V11 + 500 - 1500 c11_10 <= 0;
- V11 + 550 - 1500 c11_11 <= 0;
- V11 + 600 - 1500 c11_12 <= 0;
- V11 + 650 - 1500 c11_13 <= 0;
- V11 + 700 - 1500 c11_14 <= 0;
- V11 + 750 - 1500 c11_15 <= 0;
- V11 + 800 - 1500 c11_16 <= 0;
- V11 + 850 - 1500 c11_17 <= 0;
- V11 + 900 - 1500 c11_18 <= 0;
- V11 + 950 - 1500 c11_19 <= 0;
- V11 + 1000 - 1500 c11_20 <= 0;
- V12 + 0 - 1500 c12_0 <= 0;
- V12 + 50 - 1500 c12_1 <= 0;
- V12 + 100 - 1500 c12_2 <= 0;
- V12 + 150 - 1500 c12_3 <= 0;
- V12 + 200 - 1500 c12_4 <= 0;
- V12 + 250 - 1500 c12_5 <= 0;
- V12 + 300 - 1500 c12_6 <= 0;
- V12 + 350 - 1500 c12_7 <= 0;
- V12 + 400 - 1500 c12_8 <= 0;
- V12 + 450 - 1500 c12_9 <= 0;
- V12 + 500 - 1500 c12_10 <= 0;
- V12 + 550 - 1500 c12_11 <= 0;
- V12 + 600 - 1500 c12_12 <= 0;
- V12 + 650 - 1500 c12_13 <= 0;
```

```
- V12 + 700 - 1500 c12_14 <= 0;
- V12 + 750 - 1500 c12_15 <= 0;
- V12 + 800 - 1500 c12_16 <= 0;
- V12 + 850 - 1500 c12_17 <= 0;
- V12 + 900 - 1500 c12_18 <= 0;
- V12 + 950 - 1500 c12_19 <= 0;
- V12 + 1000 - 1500 c12_20 <= 0;
- V13 + 0 - 1500 c13_0 <= 0;
- V13 + 50 - 1500 c13_1 <= 0;
- V13 + 100 - 1500 c13_2 <= 0;
- V13 + 150 - 1500 c13_3 <= 0;
- V13 + 200 - 1500 c13_4 <= 0;
- V13 + 250 - 1500 c13_5 <= 0;
- V13 + 300 - 1500 c13_6 <= 0;
- V13 + 350 - 1500 c13_7 <= 0;
- V13 + 400 - 1500 c13_8 <= 0;
- V13 + 450 - 1500 c13_9 <= 0;
- V13 + 500 - 1500 c13_10 <= 0;
- V13 + 550 - 1500 c13_11 <= 0;
- V13 + 600 - 1500 c13_12 <= 0;
- V13 + 650 - 1500 c13_13 <= 0;
- V13 + 700 - 1500 c13_14 <= 0;
- V13 + 750 - 1500 c13_15 <= 0;
- V13 + 800 - 1500 c13_16 <= 0;
- V13 + 850 - 1500 c13_17 <= 0;
- V13 + 900 - 1500 c13_18 <= 0;
- V13 + 950 - 1500 c13_19 <= 0;
- V13 + 1000 - 1500 c13_20 <= 0;

/* constraint 12 */
c0_0 + c0_1 + c0_2 <= 2;
c1_0 + c1_1 + c1_2 <= 2;
c2_0 + c2_1 + c2_2 <= 2;
c3_0 + c3_1 + c3_2 <= 2;
c4_0 + c4_1 + c4_2 <= 2;
c5_0 + c5_1 + c5_2 <= 2;
c6_0 + c6_1 + c6_2 <= 2;
c7_0 + c7_1 + c7_2 <= 2;
c8_0 + c8_1 + c8_2 + c8_3 + c8_4 + c8_5 + c8_6 + c8_7 + c8_8 + c8_9 + c8_10 + c8_11 + c8_12 + c8_13 + c8_14 + c8_15 +
c8_16 + c8_17 + c8_18 + c8_19 + c8_20 <= 20;
c9_0 + c9_1 + c9_2 + c9_3 + c9_4 + c9_5 + c9_6 + c9_7 + c9_8 + c9_9 + c9_10 + c9_11 + c9_12 + c9_13 + c9_14 + c9_15 +
c9_16 + c9_17 + c9_18 + c9_19 + c9_20 <= 20;
c10_0 + c10_1 + c10_2 + c10_3 + c10_4 + c10_5 + c10_6 + c10_7 + c10_8 + c10_9 + c10_10 + c10_11 + c10_12 + c10_13 +
c10_14 + c10_15 + c10_16 + c10_17 + c10_18 + c10_19 + c10_20 <= 20;
c11_0 + c11_1 + c11_2 + c11_3 + c11_4 + c11_5 + c11_6 + c11_7 + c11_8 + c11_9 + c11_10 + c11_11 + c11_12 + c11_13 +
c11_14 + c11_15 + c11_16 + c11_17 + c11_18 + c11_19 + c11_20 <= 20;
c12_0 + c12_1 + c12_2 + c12_3 + c12_4 + c12_5 + c12_6 + c12_7 + c12_8 + c12_9 + c12_10 + c12_11 + c12_12 + c12_13 +
c12_14 + c12_15 + c12_16 + c12_17 + c12_18 + c12_19 + c12_20 <= 20;
c13_0 + c13_1 + c13_2 + c13_3 + c13_4 + c13_5 + c13_6 + c13_7 + c13_8 + c13_9 + c13_10 + c13_11 + c13_12 + c13_13 +
c13_14 + c13_15 + c13_16 + c13_17 + c13_18 + c13_19 + c13_20 <= 20;

/* constraint 13 */
k0_0 + c0_0 <= 1;
k0_1 + c0_1 <= 1;
k0_2 + c0_2 <= 1;
k1_0 + c1_0 <= 1;
k1_1 + c1_1 <= 1;
k1_2 + c1_2 <= 1;
k2_0 + c2_0 <= 1;
k2_1 + c2_1 <= 1;
k2_2 + c2_2 <= 1;
k3_0 + c3_0 <= 1;
k3_1 + c3_1 <= 1;
k3_2 + c3_2 <= 1;
k4_0 + c4_0 <= 1;
k4_1 + c4_1 <= 1;
k4_2 + c4_2 <= 1;
k5_0 + c5_0 <= 1;
k5_1 + c5_1 <= 1;
k5_2 + c5_2 <= 1;
k6_0 + c6_0 <= 1;
k6_1 + c6_1 <= 1;
k6_2 + c6_2 <= 1;
k7_0 + c7_0 <= 1;
k7_1 + c7_1 <= 1;
k7_2 + c7_2 <= 1;
k8_0 + c8_0 <= 1;
k8_1 + c8_1 <= 1;
k8_2 + c8_2 <= 1;
k8_3 + c8_3 <= 1;
k8_4 + c8_4 <= 1;
k8_5 + c8_5 <= 1;
k8_6 + c8_6 <= 1;
k8_7 + c8_7 <= 1;
k8_8 + c8_8 <= 1;
```

```
k8_9 + c8_9 <= 1;
k8_10 + c8_10 <= 1;
k8_11 + c8_11 <= 1;
k8_12 + c8_12 <= 1;
k8_13 + c8_13 <= 1;
k8_14 + c8_14 <= 1;
k8_15 + c8_15 <= 1;
k8_16 + c8_16 <= 1;
k8_17 + c8_17 <= 1;
k8_18 + c8_18 <= 1;
k8_19 + c8_19 <= 1;
k8_20 + c8_20 <= 1;
k9_0 + c9_0 <= 1;
k9_1 + c9_1 <= 1;
k9_2 + c9_2 <= 1;
k9_3 + c9_3 <= 1;
k9_4 + c9_4 <= 1;
k9_5 + c9_5 <= 1;
k9_6 + c9_6 <= 1;
k9_7 + c9_7 <= 1;
k9_8 + c9_8 <= 1;
k9_9 + c9_9 <= 1;
k9_10 + c9_10 <= 1;
k9_11 + c9_11 <= 1;
k9_12 + c9_12 <= 1;
k9_13 + c9_13 <= 1;
k9_14 + c9_14 <= 1;
k9_15 + c9_15 <= 1;
k9_16 + c9_16 <= 1;
k9_17 + c9_17 <= 1;
k9_18 + c9_18 <= 1;
k9_19 + c9_19 <= 1;
k9_20 + c9_20 <= 1;
k10_0 + c10_0 <= 1;
k10_1 + c10_1 <= 1;
k10_2 + c10_2 <= 1;
k10_3 + c10_3 <= 1;
k10_4 + c10_4 <= 1;
k10_5 + c10_5 <= 1;
k10_6 + c10_6 <= 1;
k10_7 + c10_7 <= 1;
k10_8 + c10_8 <= 1;
k10_9 + c10_9 <= 1;
k10_10 + c10_10 <= 1;
k10_11 + c10_11 <= 1;
k10_12 + c10_12 <= 1;
k10_13 + c10_13 <= 1;
k10_14 + c10_14 <= 1;
k10_15 + c10_15 <= 1;
k10_16 + c10_16 <= 1;
k10_17 + c10_17 <= 1;
k10_18 + c10_18 <= 1;
k10_19 + c10_19 <= 1;
k10_20 + c10_20 <= 1;
k11_0 + c11_0 <= 1;
k11_1 + c11_1 <= 1;
k11_2 + c11_2 <= 1;
k11_3 + c11_3 <= 1;
k11_4 + c11_4 <= 1;
k11_5 + c11_5 <= 1;
k11_6 + c11_6 <= 1;
k11_7 + c11_7 <= 1;
k11_8 + c11_8 <= 1;
k11_9 + c11_9 <= 1;
k11_10 + c11_10 <= 1;
k11_11 + c11_11 <= 1;
k11_12 + c11_12 <= 1;
k11_13 + c11_13 <= 1;
k11_14 + c11_14 <= 1;
k11_15 + c11_15 <= 1;
k11_16 + c11_16 <= 1;
k11_17 + c11_17 <= 1;
k11_18 + c11_18 <= 1;
k11_19 + c11_19 <= 1;
k11_20 + c11_20 <= 1;
k12_0 + c12_0 <= 1;
k12_1 + c12_1 <= 1;
k12_2 + c12_2 <= 1;
k12_3 + c12_3 <= 1;
k12_4 + c12_4 <= 1;
k12_5 + c12_5 <= 1;
k12_6 + c12_6 <= 1;
k12_7 + c12_7 <= 1;
k12_8 + c12_8 <= 1;
k12_9 + c12_9 <= 1;
```

```
k12_10 + c12_10 <= 1;
k12_11 + c12_11 <= 1;
k12_12 + c12_12 <= 1;
k12_13 + c12_13 <= 1;
k12_14 + c12_14 <= 1;
k12_15 + c12_15 <= 1;
k12_16 + c12_16 <= 1;
k12_17 + c12_17 <= 1;
k12_18 + c12_18 <= 1;
k12_19 + c12_19 <= 1;
k12_20 + c12_20 <= 1;
k13_0 + c13_0 <= 1;
k13_1 + c13_1 <= 1;
k13_2 + c13_2 <= 1;
k13_3 + c13_3 <= 1;
k13_4 + c13_4 <= 1;
k13_5 + c13_5 <= 1;
k13_6 + c13_6 <= 1;
k13_7 + c13_7 <= 1;
k13_8 + c13_8 <= 1;
k13_9 + c13_9 <= 1;
k13_10 + c13_10 <= 1;
k13_11 + c13_11 <= 1;
k13_12 + c13_12 <= 1;
k13_13 + c13_13 <= 1;
k13_14 + c13_14 <= 1;
k13_15 + c13_15 <= 1;
k13_16 + c13_16 <= 1;
k13_17 + c13_17 <= 1;
k13_18 + c13_18 <= 1;
k13_19 + c13_19 <= 1;
k13_20 + c13_20 <= 1;

/* constraint T13 */
- k0_0 - c0_0 <= -1;
- k0_1 - c0_1 <= -1;
- k0_2 - c0_2 <= -1;
- k1_0 - c1_0 <= -1;
- k1_1 - c1_1 <= -1;
- k1_2 - c1_2 <= -1;
- k2_0 - c2_0 <= -1;
- k2_1 - c2_1 <= -1;
- k2_2 - c2_2 <= -1;
- k3_0 - c3_0 <= -1;
- k3_1 - c3_1 <= -1;
- k3_2 - c3_2 <= -1;
- k4_0 - c4_0 <= -1;
- k4_1 - c4_1 <= -1;
- k4_2 - c4_2 <= -1;
- k5_0 - c5_0 <= -1;
- k5_1 - c5_1 <= -1;
- k5_2 - c5_2 <= -1;
- k6_0 - c6_0 <= -1;
- k6_1 - c6_1 <= -1;
- k6_2 - c6_2 <= -1;
- k7_0 - c7_0 <= -1;
- k7_1 - c7_1 <= -1;
- k7_2 - c7_2 <= -1;
- k8_0 - c8_0 <= -1;
- k8_1 - c8_1 <= -1;
- k8_2 - c8_2 <= -1;
- k8_3 - c8_3 <= -1;
- k8_4 - c8_4 <= -1;
- k8_5 - c8_5 <= -1;
- k8_6 - c8_6 <= -1;
- k8_7 - c8_7 <= -1;
- k8_8 - c8_8 <= -1;
- k8_9 - c8_9 <= -1;
- k8_10 - c8_10 <= -1;
- k8_11 - c8_11 <= -1;
- k8_12 - c8_12 <= -1;
- k8_13 - c8_13 <= -1;
- k8_14 - c8_14 <= -1;
- k8_15 - c8_15 <= -1;
- k8_16 - c8_16 <= -1;
- k8_17 - c8_17 <= -1;
- k8_18 - c8_18 <= -1;
- k8_19 - c8_19 <= -1;
- k8_20 - c8_20 <= -1;
- k9_0 - c9_0 <= -1;
- k9_1 - c9_1 <= -1;
- k9_2 - c9_2 <= -1;
- k9_3 - c9_3 <= -1;
- k9_4 - c9_4 <= -1;
- k9_5 - c9_5 <= -1;
```

```
- k9_6 - c9_6 <= -1;
- k9_7 - c9_7 <= -1;
- k9_8 - c9_8 <= -1;
- k9_9 - c9_9 <= -1;
- k9_10 - c9_10 <= -1;
- k9_11 - c9_11 <= -1;
- k9_12 - c9_12 <= -1;
- k9_13 - c9_13 <= -1;
- k9_14 - c9_14 <= -1;
- k9_15 - c9_15 <= -1;
- k9_16 - c9_16 <= -1;
- k9_17 - c9_17 <= -1;
- k9_18 - c9_18 <= -1;
- k9_19 - c9_19 <= -1;
- k9_20 - c9_20 <= -1;
- k10_0 - c10_0 <= -1;
- k10_1 - c10_1 <= -1;
- k10_2 - c10_2 <= -1;
- k10_3 - c10_3 <= -1;
- k10_4 - c10_4 <= -1;
- k10_5 - c10_5 <= -1;
- k10_6 - c10_6 <= -1;
- k10_7 - c10_7 <= -1;
- k10_8 - c10_8 <= -1;
- k10_9 - c10_9 <= -1;
- k10_10 - c10_10 <= -1;
- k10_11 - c10_11 <= -1;
- k10_12 - c10_12 <= -1;
- k10_13 - c10_13 <= -1;
- k10_14 - c10_14 <= -1;
- k10_15 - c10_15 <= -1;
- k10_16 - c10_16 <= -1;
- k10_17 - c10_17 <= -1;
- k10_18 - c10_18 <= -1;
- k10_19 - c10_19 <= -1;
- k10_20 - c10_20 <= -1;
- k11_0 - c11_0 <= -1;
- k11_1 - c11_1 <= -1;
- k11_2 - c11_2 <= -1;
- k11_3 - c11_3 <= -1;
- k11_4 - c11_4 <= -1;
- k11_5 - c11_5 <= -1;
- k11_6 - c11_6 <= -1;
- k11_7 - c11_7 <= -1;
- k11_8 - c11_8 <= -1;
- k11_9 - c11_9 <= -1;
- k11_10 - c11_10 <= -1;
- k11_11 - c11_11 <= -1;
- k11_12 - c11_12 <= -1;
- k11_13 - c11_13 <= -1;
- k11_14 - c11_14 <= -1;
- k11_15 - c11_15 <= -1;
- k11_16 - c11_16 <= -1;
- k11_17 - c11_17 <= -1;
- k11_18 - c11_18 <= -1;
- k11_19 - c11_19 <= -1;
- k11_20 - c11_20 <= -1;
- k12_0 - c12_0 <= -1;
- k12_1 - c12_1 <= -1;
- k12_2 - c12_2 <= -1;
- k12_3 - c12_3 <= -1;
- k12_4 - c12_4 <= -1;
- k12_5 - c12_5 <= -1;
- k12_6 - c12_6 <= -1;
- k12_7 - c12_7 <= -1;
- k12_8 - c12_8 <= -1;
- k12_9 - c12_9 <= -1;
- k12_10 - c12_10 <= -1;
- k12_11 - c12_11 <= -1;
- k12_12 - c12_12 <= -1;
- k12_13 - c12_13 <= -1;
- k12_14 - c12_14 <= -1;
- k12_15 - c12_15 <= -1;
- k12_16 - c12_16 <= -1;
- k12_17 - c12_17 <= -1;
- k12_18 - c12_18 <= -1;
- k12_19 - c12_19 <= -1;
- k12_20 - c12_20 <= -1;
- k13_0 - c13_0 <= -1;
- k13_1 - c13_1 <= -1;
- k13_2 - c13_2 <= -1;
- k13_3 - c13_3 <= -1;
- k13_4 - c13_4 <= -1;
- k13_5 - c13_5 <= -1;
- k13_6 - c13_6 <= -1;
```

```
- k13_7 - c13_7 <= -1;
- k13_8 - c13_8 <= -1;
- k13_9 - c13_9 <= -1;
- k13_10 - c13_10 <= -1;
- k13_11 - c13_11 <= -1;
- k13_12 - c13_12 <= -1;
- k13_13 - c13_13 <= -1;
- k13_14 - c13_14 <= -1;
- k13_15 - c13_15 <= -1;
- k13_16 - c13_16 <= -1;
- k13_17 - c13_17 <= -1;
- k13_18 - c13_18 <= -1;
- k13_19 - c13_19 <= -1;
- k13_20 - c13_20 <= -1;

/* compute delay on link and then path */

/* constraint 14 */
G0 - 200 k0_1 - 500 k0_2 <= 0;
G1 - 200 k1_1 - 500 k1_2 <= 0;
G2 - 200 k2_1 - 500 k2_2 <= 0;
G3 - 200 k3_1 - 500 k3_2 <= 0;
G4 - 200 k4_1 - 500 k4_2 <= 0;
G5 - 200 k5_1 - 500 k5_2 <= 0;
G6 - 200 k6_1 - 500 k6_2 <= 0;
G7 - 200 k7_1 - 500 k7_2 <= 0;
G8 - 9 k8_1 - 10 k8_2 - 11 k8_3 - 12 k8_4 - 13 k8_5 - 14 k8_6 - 15 k8_7 - 16 k8_8 - 18 k8_9 - 20 k8_10 - 22 k8_11 - 25
k8_12 - 29 k8_13 - 33 k8_14 - 40 k8_15 - 50 k8_16 - 67 k8_17 - 100 k8_18 - 200 k8_19 - 500 k8_20 <= 0;
G9 - 9 k9_1 - 10 k9_2 - 11 k9_3 - 12 k9_4 - 13 k9_5 - 14 k9_6 - 15 k9_7 - 16 k9_8 - 18 k9_9 - 20 k9_10 - 22 k9_11 - 25
k9_12 - 29 k9_13 - 33 k9_14 - 40 k9_15 - 50 k9_16 - 67 k9_17 - 100 k9_18 - 200 k9_19 - 500 k9_20 <= 0;
G10 - 9 k10_1 - 10 k10_2 - 11 k10_3 - 12 k10_4 - 13 k10_5 - 14 k10_6 - 15 k10_7 - 16 k10_8 - 18 k10_9 - 20 k10_10 - 22
k10_11 - 25 k10_12 - 29 k10_13 - 33 k10_14 - 40 k10_15 - 50 k10_16 - 67 k10_17 - 100 k10_18 - 200 k10_19 - 500 k10_20
<= 0;
G11 - 9 k11_1 - 10 k11_2 - 11 k11_3 - 12 k11_4 - 13 k11_5 - 14 k11_6 - 15 k11_7 - 16 k11_8 - 18 k11_9 - 20 k11_10 - 22
k11_11 - 25 k11_12 - 29 k11_13 - 33 k11_14 - 40 k11_15 - 50 k11_16 - 67 k11_17 - 100 k11_18 - 200 k11_19 - 500 k11_20
<= 0;
G12 - 9 k12_1 - 10 k12_2 - 11 k12_3 - 12 k12_4 - 13 k12_5 - 14 k12_6 - 15 k12_7 - 16 k12_8 - 18 k12_9 - 20 k12_10 - 22
k12_11 - 25 k12_12 - 29 k12_13 - 33 k12_14 - 40 k12_15 - 50 k12_16 - 67 k12_17 - 100 k12_18 - 200 k12_19 - 500 k12_20
<= 0;
G13 - 9 k13_1 - 10 k13_2 - 11 k13_3 - 12 k13_4 - 13 k13_5 - 14 k13_6 - 15 k13_7 - 16 k13_8 - 18 k13_9 - 20 k13_10 - 22
k13_11 - 25 k13_12 - 29 k13_13 - 33 k13_14 - 40 k13_15 - 50 k13_16 - 67 k13_17 - 100 k13_18 - 200 k13_19 - 500 k13_20
<= 0;

/* constraint T14 */
 - G0 + 200 k0_1 + 500 k0_2 <= 0;
 - G1 + 200 k1_1 + 500 k1_2 <= 0;
 - G2 + 200 k2_1 + 500 k2_2 <= 0;
 - G3 + 200 k3_1 + 500 k3_2 <= 0;
 - G4 + 200 k4_1 + 500 k4_2 <= 0;
 - G5 + 200 k5_1 + 500 k5_2 <= 0;
 - G6 + 200 k6_1 + 500 k6_2 <= 0;
 - G7 + 200 k7_1 + 500 k7_2 <= 0;
 - G8 + 9 k8_1 + 10 k8_2 + 11 k8_3 + 12 k8_4 + 13 k8_5 + 14 k8_6 + 15 k8_7 + 16 k8_8 + 18 k8_9 + 20 k8_10 + 22 k8_11 +
25 k8_12 + 29 k8_13 + 33 k8_14 + 40 k8_15 + 50 k8_16 + 67 k8_17 + 100 k8_18 + 200 k8_19 + 500 k8_20 <= 0;
 - G9 + 9 k9_1 + 10 k9_2 + 11 k9_3 + 12 k9_4 + 13 k9_5 + 14 k9_6 + 15 k9_7 + 16 k9_8 + 18 k9_9 + 20 k9_10 + 22 k9_11 +
25 k9_12 + 29 k9_13 + 33 k9_14 + 40 k9_15 + 50 k9_16 + 67 k9_17 + 100 k9_18 + 200 k9_19 + 500 k9_20 <= 0;
 - G10 + 9 k10_1 + 10 k10_2 + 11 k10_3 + 12 k10_4 + 13 k10_5 + 14 k10_6 + 15 k10_7 + 16 k10_8 + 18 k10_9 + 20 k10_10 +
22 k10_11 + 25 k10_12 + 29 k10_13 + 33 k10_14 + 40 k10_15 + 50 k10_16 + 67 k10_17 + 100 k10_18 + 200 k10_19 + 500
k10_20 <= 0;
 - G11 + 9 k11_1 + 10 k11_2 + 11 k11_3 + 12 k11_4 + 13 k11_5 + 14 k11_6 + 15 k11_7 + 16 k11_8 + 18 k11_9 + 20 k11_10 +
22 k11_11 + 25 k11_12 + 29 k11_13 + 33 k11_14 + 40 k11_15 + 50 k11_16 + 67 k11_17 + 100 k11_18 + 200 k11_19 + 500
k11_20 <= 0;
 - G12 + 9 k12_1 + 10 k12_2 + 11 k12_3 + 12 k12_4 + 13 k12_5 + 14 k12_6 + 15 k12_7 + 16 k12_8 + 18 k12_9 + 20 k12_10 +
22 k12_11 + 25 k12_12 + 29 k12_13 + 33 k12_14 + 40 k12_15 + 50 k12_16 + 67 k12_17 + 100 k12_18 + 200 k12_19 + 500
k12_20 <= 0;
 - G13 + 9 k13_1 + 10 k13_2 + 11 k13_3 + 12 k13_4 + 13 k13_5 + 14 k13_6 + 15 k13_7 + 16 k13_8 + 18 k13_9 + 20 k13_10 +
22 k13_11 + 25 k13_12 + 29 k13_13 + 33 k13_14 + 40 k13_15 + 50 k13_16 + 67 k13_17 + 100 k13_18 + 200 k13_19 + 500
k13_20 <= 0;

/* constraint 15 */
H0 - G0 <= 0;
H1 - G1 - G2 <= 0;
H2 - G1 <= 0;
H3 - G0 - G2 <= 0;
H4 - G5 <= 0;
H5 - G6 - G7 <= 0;
H6 - G6 <= 0;
H7 - G5 - G7 <= 0;
H8 - G0 <= 0;
H9 - G1 - G2 <= 0;
H10 - G1 <= 0;
H11 - G0 - G2 <= 0;
H12 - G5 <= 0;
H13 - G6 - G7 <= 0;
H14 - G6 <= 0;
```

```
H15 - G5 - G7 <= 0;
H16 - G8 <= 0;
H17 - G9 <= 0;
H18 - G10 <= 0;
H19 - G11 <= 0;
H20 - G12 <= 0;
H21 - G13 <= 0;

/* constraint T15 */
 - H0 + G0 <= 0;
 - H1 + G1 + G2 <= 0;
 - H2 + G1 <= 0;
 - H3 + G0 + G2 <= 0;
 - H4 + G5 <= 0;
 - H5 + G6 + G7 <= 0;
 - H6 + G6 <= 0;
 - H7 + G5 + G7 <= 0;
 - H8 + G0 <= 0;
 - H9 + G1 + G2 <= 0;
 - H10 + G1 <= 0;
 - H11 + G0 + G2 <= 0;
 - H12 + G5 <= 0;
 - H13 + G6 + G7 <= 0;
 - H14 + G6 <= 0;
 - H15 + G5 + G7 <= 0;
 - H16 + G8 <= 0;
 - H17 + G9 <= 0;
 - H18 + G10 <= 0;
 - H19 + G11 <= 0;
 - H20 + G12 <= 0;
 - H21 + G13 <= 0;

/* meeting delay requirements */

/* constraint 16 */
 - Y16_0 - r0_0_0_16 <= -1;
 - Y16_2 - r0_0_2_16 <= -1;
 - Y2_2 - r0_2_2_2 <= -1;
 - Y3_2 - r0_2_2_3 <= -1;
 - Y8_0 - r1_0_0_8 <= -1;
 - Y9_0 - r1_0_0_9 <= -1;
 - Y8_2 - r1_0_2_8 <= -1;
 - Y9_2 - r1_0_2_9 <= -1;
 - Y10_0 - r2_0_0_10 <= -1;
 - Y11_0 - r2_0_0_11 <= -1;
 - Y10_2 - r2_0_2_10 <= -1;
 - Y11_2 - r2_0_2_11 <= -1;
 - Y18_2 - r2_2_2_18 <= -1;
 - Y12_1 - r3_5_1_12 <= -1;
 - Y13_1 - r3_5_1_13 <= -1;
 - Y14_1 - r4_5_1_14 <= -1;
 - Y15_1 - r4_5_1_15 <= -1;
 - Y21_1 - r5_5_1_21 <= -1;

/* constraint 17 */
Y16_0 + r0_0_0_16 <= 1;
Y16_2 + r0_0_2_16 <= 1;
Y2_2 + r0_2_2_2 <= 1;
Y3_2 + r0_2_2_3 <= 1;
Y8_0 + r1_0_0_8 <= 1;
Y9_0 + r1_0_0_9 <= 1;
Y8_2 + r1_0_2_8 <= 1;
Y9_2 + r1_0_2_9 <= 1;
Y10_0 + r2_0_0_10 <= 1;
Y11_0 + r2_0_0_11 <= 1;
Y10_2 + r2_0_2_10 <= 1;
Y11_2 + r2_0_2_11 <= 1;
Y18_2 + r2_2_2_18 <= 1;
Y12_1 + r3_5_1_12 <= 1;
Y13_1 + r3_5_1_13 <= 1;
Y14_1 + r4_5_1_14 <= 1;
Y15_1 + r4_5_1_15 <= 1;
Y21_1 + r5_5_1_21 <= 1;

/* constraint 18 */

 H16 - 1500 r0_0_0_16 <= 1000;
 H16 - 1500 r0_0_2_16 <= 15;
 H2 - 1500 r0_2_2_2 <= 1000;
 H3 - 1500 r0_2_2_3 <= 1000;
 H8 - 1500 r1_0_0_8 <= 1000;
 H9 - 1500 r1_0_0_9 <= 1000;
 H8 - 1500 r1_0_2_8 <= 15;
 H9 - 1500 r1_0_2_9 <= 15;
 H10 - 1500 r2_0_0_10 <= 1000;
```

```
H11 - 1500 r2_0_0_11 <= 1000;
H10 - 1500 r2_0_2_10 <= 15;
H11 - 1500 r2_0_2_11 <= 15;
H18 - 1500 r2_2_2_18 <= 1000;
H12 - 1500 r3_5_1_12 <= 1000;
H13 - 1500 r3_5_1_13 <= 1000;
H14 - 1500 r4_5_1_14 <= 1000;
H15 - 1500 r4_5_1_15 <= 1000;
H21 - 1500 r5_5_1_21 <= 1000;
/* constraint 19 */
- 11000 Y16_0 - 11000 Y8_0 - 11000 Y9_0 - 11000 Y10_0 - 11000 Y11_0 <= - 137;
- 11000 Y16_1 - 11000 Y8_1 - 11000 Y9_1 - 11000 Y10_1 - 11000 Y11_1 <= - 0;
- 11000 Y16_2 - 11000 Y8_2 - 11000 Y9_2 - 11000 Y10_2 - 11000 Y11_2 <= - 200;
- 11000 Y0_0 - 11000 Y1_0 - 11000 Y17_0 <= - 0;
- 11000 Y0_1 - 11000 Y1_1 - 11000 Y17_1 <= - 0;
- 11000 Y0_2 - 11000 Y1_2 - 11000 Y17_2 <= - 0;
- 11000 Y2_0 - 11000 Y3_0 - 11000 Y18_0 <= - 0;
- 11000 Y2_1 - 11000 Y3_1 - 11000 Y18_1 <= - 0;
- 11000 Y2_2 - 11000 Y3_2 - 11000 Y18_2 <= - 25;
- 11000 Y19_0 - 11000 Y4_0 - 11000 Y5_0 <= - 0;
- 11000 Y19_1 - 11000 Y4_1 - 11000 Y5_1 <= - 0;
- 11000 Y19_2 - 11000 Y4_2 - 11000 Y5_2 <= - 0;
- 11000 Y20_0 - 11000 Y6_0 - 11000 Y7_0 <= - 0;
- 11000 Y20_1 - 11000 Y6_1 - 11000 Y7_1 <= - 0;
- 11000 Y20_2 - 11000 Y6_2 - 11000 Y7_2 <= - 0;
- 11000 Y12_0 - 11000 Y13_0 - 11000 Y14_0 - 11000 Y15_0 - 11000 Y21_0 <= - 0;
- 11000 Y12_1 - 11000 Y13_1 - 11000 Y14_1 - 11000 Y15_1 - 11000 Y21_1 <= - 180;
- 11000 Y12_2 - 11000 Y13_2 - 11000 Y14_2 - 11000 Y15_2 - 11000 Y21_2 <= - 0;

/* constraint 20 */
+ Y16_0 + Y8_0 + Y9_0 + Y10_0 + Y11_0 <= 137;
+ Y16_1 + Y8_1 + Y9_1 + Y10_1 + Y11_1 <= 0;
+ Y16_2 + Y8_2 + Y9_2 + Y10_2 + Y11_2 <= 200;
+ Y0_0 + Y1_0 + Y17_0 <= 0;
+ Y0_1 + Y1_1 + Y17_1 <= 0;
+ Y0_2 + Y1_2 + Y17_2 <= 0;
+ Y2_0 + Y3_0 + Y18_0 <= 0;
+ Y2_1 + Y3_1 + Y18_1 <= 0;
+ Y2_2 + Y3_2 + Y18_2 <= 25;
+ Y19_0 + Y4_0 + Y5_0 <= 0;
+ Y19_1 + Y4_1 + Y5_1 <= 0;
+ Y19_2 + Y4_2 + Y5_2 <= 0;
+ Y20_0 + Y6_0 + Y7_0 <= 0;
+ Y20_1 + Y6_1 + Y7_1 <= 0;
+ Y20_2 + Y6_2 + Y7_2 <= 0;
+ Y12_0 + Y13_0 + Y14_0 + Y15_0 + Y21_0 <= 0;
+ Y12_1 + Y13_1 + Y14_1 + Y15_1 + Y21_1 <= 180;
+ Y12_2 + Y13_2 + Y14_2 + Y15_2 + Y21_2 <= 0;



0 <= X0_0 <= 1;
0 <= X0_1 <= 1;
0 <= X0_2 <= 1;
0 <= X1_0 <= 1;
0 <= X1_1 <= 1;
0 <= X1_2 <= 1;
0 <= X2_0 <= 1;
0 <= X2_1 <= 1;
0 <= X2_2 <= 1;
0 <= X3_0 <= 1;
0 <= X3_1 <= 1;
0 <= X3_2 <= 1;
0 <= X4_0 <= 1;
0 <= X4_1 <= 1;
0 <= X4_2 <= 1;
0 <= X5_0 <= 1;
0 <= X5_1 <= 1;
0 <= X5_2 <= 1;
0 <= U0<= 1;
0 <= U1<= 1;
0 <= U2<= 1;
0 <= U3<= 1;
0 <= U4<= 1;
0 <= U5<= 1;
0 <= Y0_0 <= 1;
0 <= Y0_1 <= 1;
0 <= Y0_2 <= 1;
0 <= Y1_0 <= 1;
0 <= Y1_1 <= 1;
0 <= Y1_2 <= 1;
0 <= Y2_0 <= 1;
0 <= Y2_1 <= 1;
0 <= Y2_2 <= 1;
```

```
0 <= Y3_0 <= 1;
0 <= Y3_1 <= 1;
0 <= Y3_2 <= 1;
0 <= Y4_0 <= 1;
0 <= Y4_1 <= 1;
0 <= Y4_2 <= 1;
0 <= Y5_0 <= 1;
0 <= Y5_1 <= 1;
0 <= Y5_2 <= 1;
0 <= Y6_0 <= 1;
0 <= Y6_1 <= 1;
0 <= Y6_2 <= 1;
0 <= Y7_0 <= 1;
0 <= Y7_1 <= 1;
0 <= Y7_2 <= 1;
0 <= Y8_0 <= 1;
0 <= Y8_1 <= 1;
0 <= Y8_2 <= 1;
0 <= Y9_0 <= 1;
0 <= Y9_1 <= 1;
0 <= Y9_2 <= 1;
0 <= Y10_0 <= 1;
0 <= Y10_1 <= 1;
0 <= Y10_2 <= 1;
0 <= Y11_0 <= 1;
0 <= Y11_1 <= 1;
0 <= Y11_2 <= 1;
0 <= Y12_0 <= 1;
0 <= Y12_1 <= 1;
0 <= Y12_2 <= 1;
0 <= Y13_0 <= 1;
0 <= Y13_1 <= 1;
0 <= Y13_2 <= 1;
0 <= Y14_0 <= 1;
0 <= Y14_1 <= 1;
0 <= Y14_2 <= 1;
0 <= Y15_0 <= 1;
0 <= Y15_1 <= 1;
0 <= Y15_2 <= 1;
0 <= Y16_0 <= 1;
0 <= Y16_1 <= 1;
0 <= Y16_2 <= 1;
0 <= Y17_0 <= 1;
0 <= Y17_1 <= 1;
0 <= Y17_2 <= 1;
0 <= Y18_0 <= 1;
0 <= Y18_1 <= 1;
0 <= Y18_2 <= 1;
0 <= Y19_0 <= 1;
0 <= Y19_1 <= 1;
0 <= Y19_2 <= 1;
0 <= Y20_0 <= 1;
0 <= Y20_1 <= 1;
0 <= Y20_2 <= 1;
0 <= Y21_0 <= 1;
0 <= Y21_1 <= 1;
0 <= Y21_2 <= 1;
0 <= c0_0 <= 1;
0 <= c0_1 <= 1;
0 <= c0_2 <= 1;
0 <= c1_0 <= 1;
0 <= c1_1 <= 1;
0 <= c1_2 <= 1;
0 <= c2_0 <= 1;
0 <= c2_1 <= 1;
0 <= c2_2 <= 1;
0 <= c3_0 <= 1;
0 <= c3_1 <= 1;
0 <= c3_2 <= 1;
0 <= c4_0 <= 1;
0 <= c4_1 <= 1;
0 <= c4_2 <= 1;
0 <= c5_0 <= 1;
0 <= c5_1 <= 1;
0 <= c5_2 <= 1;
0 <= c6_0 <= 1;
0 <= c6_1 <= 1;
0 <= c6_2 <= 1;
0 <= c7_0 <= 1;
0 <= c7_1 <= 1;
0 <= c7_2 <= 1;
0 <= c8_0 <= 1;
0 <= c8_1 <= 1;
0 <= c8_2 <= 1;
0 <= c8_3 <= 1;
```

```
0 <= c8_4 <= 1;
0 <= c8_5 <= 1;
0 <= c8_6 <= 1;
0 <= c8_7 <= 1;
0 <= c8_8 <= 1;
0 <= c8_9 <= 1;
0 <= c8_10 <= 1;
0 <= c8_11 <= 1;
0 <= c8_12 <= 1;
0 <= c8_13 <= 1;
0 <= c8_14 <= 1;
0 <= c8_15 <= 1;
0 <= c8_16 <= 1;
0 <= c8_17 <= 1;
0 <= c8_18 <= 1;
0 <= c8_19 <= 1;
0 <= c8_20 <= 1;
0 <= c9_0 <= 1;
0 <= c9_1 <= 1;
0 <= c9_2 <= 1;
0 <= c9_3 <= 1;
0 <= c9_4 <= 1;
0 <= c9_5 <= 1;
0 <= c9_6 <= 1;
0 <= c9_7 <= 1;
0 <= c9_8 <= 1;
0 <= c9_9 <= 1;
0 <= c9_10 <= 1;
0 <= c9_11 <= 1;
0 <= c9_12 <= 1;
0 <= c9_13 <= 1;
0 <= c9_14 <= 1;
0 <= c9_15 <= 1;
0 <= c9_16 <= 1;
0 <= c9_17 <= 1;
0 <= c9_18 <= 1;
0 <= c9_19 <= 1;
0 <= c9_20 <= 1;
0 <= c10_0 <= 1;
0 <= c10_1 <= 1;
0 <= c10_2 <= 1;
0 <= c10_3 <= 1;
0 <= c10_4 <= 1;
0 <= c10_5 <= 1;
0 <= c10_6 <= 1;
0 <= c10_7 <= 1;
0 <= c10_8 <= 1;
0 <= c10_9 <= 1;
0 <= c10_10 <= 1;
0 <= c10_11 <= 1;
0 <= c10_12 <= 1;
0 <= c10_13 <= 1;
0 <= c10_14 <= 1;
0 <= c10_15 <= 1;
0 <= c10_16 <= 1;
0 <= c10_17 <= 1;
0 <= c10_18 <= 1;
0 <= c10_19 <= 1;
0 <= c10_20 <= 1;
0 <= c11_0 <= 1;
0 <= c11_1 <= 1;
0 <= c11_2 <= 1;
0 <= c11_3 <= 1;
0 <= c11_4 <= 1;
0 <= c11_5 <= 1;
0 <= c11_6 <= 1;
0 <= c11_7 <= 1;
0 <= c11_8 <= 1;
0 <= c11_9 <= 1;
0 <= c11_10 <= 1;
0 <= c11_11 <= 1;
0 <= c11_12 <= 1;
0 <= c11_13 <= 1;
0 <= c11_14 <= 1;
0 <= c11_15 <= 1;
0 <= c11_16 <= 1;
0 <= c11_17 <= 1;
0 <= c11_18 <= 1;
0 <= c11_19 <= 1;
0 <= c11_20 <= 1;
0 <= c12_0 <= 1;
0 <= c12_1 <= 1;
0 <= c12_2 <= 1;
0 <= c12_3 <= 1;
0 <= c12_4 <= 1;
```

```
0 <= c12_5 <= 1;
0 <= c12_6 <= 1;
0 <= c12_7 <= 1;
0 <= c12_8 <= 1;
0 <= c12_9 <= 1;
0 <= c12_10 <= 1;
0 <= c12_11 <= 1;
0 <= c12_12 <= 1;
0 <= c12_13 <= 1;
0 <= c12_14 <= 1;
0 <= c12_15 <= 1;
0 <= c12_16 <= 1;
0 <= c12_17 <= 1;
0 <= c12_18 <= 1;
0 <= c12_19 <= 1;
0 <= c12_20 <= 1;
0 <= c13_0 <= 1;
0 <= c13_1 <= 1;
0 <= c13_2 <= 1;
0 <= c13_3 <= 1;
0 <= c13_4 <= 1;
0 <= c13_5 <= 1;
0 <= c13_6 <= 1;
0 <= c13_7 <= 1;
0 <= c13_8 <= 1;
0 <= c13_9 <= 1;
0 <= c13_10 <= 1;
0 <= c13_11 <= 1;
0 <= c13_12 <= 1;
0 <= c13_13 <= 1;
0 <= c13_14 <= 1;
0 <= c13_15 <= 1;
0 <= c13_16 <= 1;
0 <= c13_17 <= 1;
0 <= c13_18 <= 1;
0 <= c13_19 <= 1;
0 <= c13_20 <= 1;
0 <= k0_0 <= 1;
0 <= k0_1 <= 1;
0 <= k0_2 <= 1;
0 <= k1_0 <= 1;
0 <= k1_1 <= 1;
0 <= k1_2 <= 1;
0 <= k2_0 <= 1;
0 <= k2_1 <= 1;
0 <= k2_2 <= 1;
0 <= k3_0 <= 1;
0 <= k3_1 <= 1;
0 <= k3_2 <= 1;
0 <= k4_0 <= 1;
0 <= k4_1 <= 1;
0 <= k4_2 <= 1;
0 <= k5_0 <= 1;
0 <= k5_1 <= 1;
0 <= k5_2 <= 1;
0 <= k6_0 <= 1;
0 <= k6_1 <= 1;
0 <= k6_2 <= 1;
0 <= k7_0 <= 1;
0 <= k7_1 <= 1;
0 <= k7_2 <= 1;
0 <= k8_0 <= 1;
0 <= k8_1 <= 1;
0 <= k8_2 <= 1;
0 <= k8_3 <= 1;
0 <= k8_4 <= 1;
0 <= k8_5 <= 1;
0 <= k8_6 <= 1;
0 <= k8_7 <= 1;
0 <= k8_8 <= 1;
0 <= k8_9 <= 1;
0 <= k8_10 <= 1;
0 <= k8_11 <= 1;
0 <= k8_12 <= 1;
0 <= k8_13 <= 1;
0 <= k8_14 <= 1;
0 <= k8_15 <= 1;
0 <= k8_16 <= 1;
0 <= k8_17 <= 1;
0 <= k8_18 <= 1;
0 <= k8_19 <= 1;
0 <= k8_20 <= 1;
0 <= k9_0 <= 1;
0 <= k9_1 <= 1;
0 <= k9_2 <= 1;
```

```
0 <= k9_3 <= 1;
0 <= k9_4 <= 1;
0 <= k9_5 <= 1;
0 <= k9_6 <= 1;
0 <= k9_7 <= 1;
0 <= k9_8 <= 1;
0 <= k9_9 <= 1;
0 <= k9_10 <= 1;
0 <= k9_11 <= 1;
0 <= k9_12 <= 1;
0 <= k9_13 <= 1;
0 <= k9_14 <= 1;
0 <= k9_15 <= 1;
0 <= k9_16 <= 1;
0 <= k9_17 <= 1;
0 <= k9_18 <= 1;
0 <= k9_19 <= 1;
0 <= k9_20 <= 1;
0 <= k10_0 <= 1;
0 <= k10_1 <= 1;
0 <= k10_2 <= 1;
0 <= k10_3 <= 1;
0 <= k10_4 <= 1;
0 <= k10_5 <= 1;
0 <= k10_6 <= 1;
0 <= k10_7 <= 1;
0 <= k10_8 <= 1;
0 <= k10_9 <= 1;
0 <= k10_10 <= 1;
0 <= k10_11 <= 1;
0 <= k10_12 <= 1;
0 <= k10_13 <= 1;
0 <= k10_14 <= 1;
0 <= k10_15 <= 1;
0 <= k10_16 <= 1;
0 <= k10_17 <= 1;
0 <= k10_18 <= 1;
0 <= k10_19 <= 1;
0 <= k10_20 <= 1;
0 <= k11_0 <= 1;
0 <= k11_1 <= 1;
0 <= k11_2 <= 1;
0 <= k11_3 <= 1;
0 <= k11_4 <= 1;
0 <= k11_5 <= 1;
0 <= k11_6 <= 1;
0 <= k11_7 <= 1;
0 <= k11_8 <= 1;
0 <= k11_9 <= 1;
0 <= k11_10 <= 1;
0 <= k11_11 <= 1;
0 <= k11_12 <= 1;
0 <= k11_13 <= 1;
0 <= k11_14 <= 1;
0 <= k11_15 <= 1;
0 <= k11_16 <= 1;
0 <= k11_17 <= 1;
0 <= k11_18 <= 1;
0 <= k11_19 <= 1;
0 <= k11_20 <= 1;
0 <= k12_0 <= 1;
0 <= k12_1 <= 1;
0 <= k12_2 <= 1;
0 <= k12_3 <= 1;
0 <= k12_4 <= 1;
0 <= k12_5 <= 1;
0 <= k12_6 <= 1;
0 <= k12_7 <= 1;
0 <= k12_8 <= 1;
0 <= k12_9 <= 1;
0 <= k12_10 <= 1;
0 <= k12_11 <= 1;
0 <= k12_12 <= 1;
0 <= k12_13 <= 1;
0 <= k12_14 <= 1;
0 <= k12_15 <= 1;
0 <= k12_16 <= 1;
0 <= k12_17 <= 1;
0 <= k12_18 <= 1;
0 <= k12_19 <= 1;
0 <= k12_20 <= 1;
0 <= k13_0 <= 1;
0 <= k13_1 <= 1;
0 <= k13_2 <= 1;
0 <= k13_3 <= 1;
```

```
0 <= k13_4 <= 1;
0 <= k13_5 <= 1;
0 <= k13_6 <= 1;
0 <= k13_7 <= 1;
0 <= k13_8 <= 1;
0 <= k13_9 <= 1;
0 <= k13_10 <= 1;
0 <= k13_11 <= 1;
0 <= k13_12 <= 1;
0 <= k13_13 <= 1;
0 <= k13_14 <= 1;
0 <= k13_15 <= 1;
0 <= k13_16 <= 1;
0 <= k13_17 <= 1;
0 <= k13_18 <= 1;
0 <= k13_19 <= 1;
0 <= k13_20 <= 1;
0 <= r0_0_0_16 <= 1;
0 <= r0_0_2_16 <= 1;
0 <= r0_2_2_2 <= 1;
0 <= r0_2_2_3 <= 1;
0 <= r1_0_0_8 <= 1;
0 <= r1_0_0_9 <= 1;
0 <= r1_0_2_8 <= 1;
0 <= r1_0_2_9 <= 1;
0 <= r2_0_0_10 <= 1;
0 <= r2_0_0_11 <= 1;
0 <= r2_0_2_10 <= 1;
0 <= r2_0_2_11 <= 1;
0 <= r2_2_2_18 <= 1;
0 <= r3_5_1_12 <= 1;
0 <= r3_5_1_13 <= 1;
0 <= r4_5_1_14 <= 1;
0 <= r4_5_1_15 <= 1;
0 <= r5_5_1_21 <= 1;
```

```
int X0_0;
int X0_1;
int X0_2;
int X1_0;
int X1_1;
int X1_2;
int X2_0;
int X2_1;
int X2_2;
int X3_0;
int X3_1;
int X3_2;
int X4_0;
int X4_1;
int X4_2;
int X5_0;
int X5_1;
int X5_2;
int U0;
int U1;
int U2;
int U3;
int U4;
int U5;
int V0;
int V1;
int V2;
int V3;
int V4;
int V5;
int V6;
int V7;
int V8;
int V9;
int V10;
int V11;
int V12;
int V13;
int G0;
int G1;
int G2;
int G3;
int G4;
```

```
int G5;
int G6;
int G7;
int G8;
int G9;
int G10;
int G11;
int G12;
int G13;
int H0;
int H1;
int H2;
int H3;
int H4;
int H5;
int H6;
int H7;
int H8;
int H9;
int H10;
int H11;
int H12;
int H13;
int H14;
int H15;
int H16;
int H17;
int H18;
int H19;
int H20;
int H21;
int Y0_0;
int Y0_1;
int Y0_2;
int Y1_0;
int Y1_1;
int Y1_2;
int Y2_0;
int Y2_1;
int Y2_2;
int Y3_0;
int Y3_1;
int Y3_2;
int Y4_0;
int Y4_1;
int Y4_2;
int Y5_0;
int Y5_1;
int Y5_2;
int Y6_0;
int Y6_1;
int Y6_2;
int Y7_0;
int Y7_1;
int Y7_2;
int Y8_0;
int Y8_1;
int Y8_2;
int Y9_0;
int Y9_1;
int Y9_2;
int Y10_0;
int Y10_1;
int Y10_2;
int Y11_0;
int Y11_1;
int Y11_2;
int Y12_0;
int Y12_1;
int Y12_2;
int Y13_0;
int Y13_1;
int Y13_2;
int Y14_0;
int Y14_1;
int Y14_2;
int Y15_0;
int Y15_1;
int Y15_2;
int Y16_0;
int Y16_1;
int Y16_2;
int Y17_0;
int Y17_1;
int Y17_2;
```

```
int Y18_0;
int Y18_1;
int Y18_2;
int Y19_0;
int Y19_1;
int Y19_2;
int Y20_0;
int Y20_1;
int Y20_2;
int Y21_0;
int Y21_1;
int Y21_2;
int a0;
int a1;
int a2;
int a3;
int a4;
int a5;
int a6;
int a7;
int a8;
int a9;
int a10;
int a11;
int a12;
int a13;
int c0_0;
int c0_1;
int c0_2;
int c1_0;
int c1_1;
int c1_2;
int c2_0;
int c2_1;
int c2_2;
int c3_0;
int c3_1;
int c3_2;
int c4_0;
int c4_1;
int c4_2;
int c5_0;
int c5_1;
int c5_2;
int c6_0;
int c6_1;
int c6_2;
int c7_0;
int c7_1;
int c7_2;
int c8_0;
int c8_1;
int c8_2;
int c8_3;
int c8_4;
int c8_5;
int c8_6;
int c8_7;
int c8_8;
int c8_9;
int c8_10;
int c8_11;
int c8_12;
int c8_13;
int c8_14;
int c8_15;
int c8_16;
int c8_17;
int c8_18;
int c8_19;
int c8_20;
int c9_0;
int c9_1;
int c9_2;
int c9_3;
int c9_4;
int c9_5;
int c9_6;
int c9_7;
int c9_8;
int c9_9;
int c9_10;
int c9_11;
int c9_12;
int c9_13;
```

```
int c9_14;
int c9_15;
int c9_16;
int c9_17;
int c9_18;
int c9_19;
int c9_20;
int c10_0;
int c10_1;
int c10_2;
int c10_3;
int c10_4;
int c10_5;
int c10_6;
int c10_7;
int c10_8;
int c10_9;
int c10_10;
int c10_11;
int c10_12;
int c10_13;
int c10_14;
int c10_15;
int c10_16;
int c10_17;
int c10_18;
int c10_19;
int c10_20;
int c11_0;
int c11_1;
int c11_2;
int c11_3;
int c11_4;
int c11_5;
int c11_6;
int c11_7;
int c11_8;
int c11_9;
int c11_10;
int c11_11;
int c11_12;
int c11_13;
int c11_14;
int c11_15;
int c11_16;
int c11_17;
int c11_18;
int c11_19;
int c11_20;
int c12_0;
int c12_1;
int c12_2;
int c12_3;
int c12_4;
int c12_5;
int c12_6;
int c12_7;
int c12_8;
int c12_9;
int c12_10;
int c12_11;
int c12_12;
int c12_13;
int c12_14;
int c12_15;
int c12_16;
int c12_17;
int c12_18;
int c12_19;
int c12_20;
int c13_0;
int c13_1;
int c13_2;
int c13_3;
int c13_4;
int c13_5;
int c13_6;
int c13_7;
int c13_8;
int c13_9;
int c13_10;
int c13_11;
int c13_12;
int c13_13;
int c13_14;
```

```
int c13_15;
int c13_16;
int c13_17;
int c13_18;
int c13_19;
int c13_20;
int k0_0;
int k0_1;
int k0_2;
int k1_0;
int k1_1;
int k1_2;
int k2_0;
int k2_1;
int k2_2;
int k3_0;
int k3_1;
int k3_2;
int k4_0;
int k4_1;
int k4_2;
int k5_0;
int k5_1;
int k5_2;
int k6_0;
int k6_1;
int k6_2;
int k7_0;
int k7_1;
int k7_2;
int k8_0;
int k8_1;
int k8_2;
int k8_3;
int k8_4;
int k8_5;
int k8_6;
int k8_7;
int k8_8;
int k8_9;
int k8_10;
int k8_11;
int k8_12;
int k8_13;
int k8_14;
int k8_15;
int k8_16;
int k8_17;
int k8_18;
int k8_19;
int k8_20;
int k9_0;
int k9_1;
int k9_2;
int k9_3;
int k9_4;
int k9_5;
int k9_6;
int k9_7;
int k9_8;
int k9_9;
int k9_10;
int k9_11;
int k9_12;
int k9_13;
int k9_14;
int k9_15;
int k9_16;
int k9_17;
int k9_18;
int k9_19;
int k9_20;
int k10_0;
int k10_1;
int k10_2;
int k10_3;
int k10_4;
int k10_5;
int k10_6;
int k10_7;
int k10_8;
int k10_9;
int k10_10;
int k10_11;
int k10_12;
```

```
int k10_13;
int k10_14;
int k10_15;
int k10_16;
int k10_17;
int k10_18;
int k10_19;
int k10_20;
int k11_0;
int k11_1;
int k11_2;
int k11_3;
int k11_4;
int k11_5;
int k11_6;
int k11_7;
int k11_8;
int k11_9;
int k11_10;
int k11_11;
int k11_12;
int k11_13;
int k11_14;
int k11_15;
int k11_16;
int k11_17;
int k11_18;
int k11_19;
int k11_20;
int k12_0;
int k12_1;
int k12_2;
int k12_3;
int k12_4;
int k12_5;
int k12_6;
int k12_7;
int k12_8;
int k12_9;
int k12_10;
int k12_11;
int k12_12;
int k12_13;
int k12_14;
int k12_15;
int k12_16;
int k12_17;
int k12_18;
int k12_19;
int k12_20;
int k13_0;
int k13_1;
int k13_2;
int k13_3;
int k13_4;
int k13_5;
int k13_6;
int k13_7;
int k13_8;
int k13_9;
int k13_10;
int k13_11;
int k13_12;
int k13_13;
int k13_14;
int k13_15;
int k13_16;
int k13_17;
int k13_18;
int k13_19;
int k13_20;
int r0_0_0_16;
int r0_0_2_16;
int r0_2_2_2;
int r0_2_2_3;
int r1_0_0_8;
int r1_0_0_9;
int r1_0_2_8;
int r1_0_2_9;
int r2_0_0_10;
int r2_0_0_11;
int r2_0_2_10;
int r2_0_2_11;
int r2_2_2_18;
int r3_5_1_12;
```

```
int r3_5_1_13;
int r4_5_1_14;
int r4_5_1_15;
int r5_5_1_21;
int Z0_0;
int Z0_1;
int Z0_2;
int Z1_0;
int Z1_1;
int Z1_2;
int Z2_0;
int Z2_1;
int Z2_2;
int Z3_0;
int Z3_1;
int Z3_2;
int Z4_0;
int Z4_1;
int Z4_2;
int Z5_0;
int Z5_1;
int Z5_2;
int Z6_0;
int Z6_1;
int Z6_2;
int Z7_0;
int Z7_1;
int Z7_2;
int Z8_0;
int Z8_1;
int Z8_2;
int Z9_0;
int Z9_1;
int Z9_2;
int Z10_0;
int Z10_1;
int Z10_2;
int Z11_0;
int Z11_1;
int Z11_2;
int Z12_0;
int Z12_1;
int Z12_2;
int Z13_0;
int Z13_1;
int Z13_2;
int Z14_0;
int Z14_1;
int Z14_2;
int Z15_0;
int Z15_1;
int Z15_2;
int Z16_0;
int Z16_1;
int Z16_2;
int Z17_0;
int Z17_1;
int Z17_2;
int Z18_0;
int Z18_1;
int Z18_2;
int Z19_0;
int Z19_1;
int Z19_2;
int Z20_0;
int Z20_1;
int Z20_2;
int Z21_0;
int Z21_1;
int Z21_2;
```

Sample Lagrange Relaxation Output File

```
-------------------------------------------------------------
currentRelaxedValue = 152.6875 best relaxed sol = 324.9375
step = 0.0625    total iterations = 45
number of constraints not relaxed = 876
original model has 1038 constraints
therefore we are relaxing 162 constraints


Elapsed Time: 45.25


 The lagrange multipliers are:
lambda 1  = 0.0
lambda 2  = 0.0
lambda 3  = 0.0
lambda 4  = 0.0
lambda 5  = 0.0
lambda 6  = 0.0625
lambda 7  = -17.75
lambda 8  = 0.0
lambda 9  = -17.75
lambda 10  = 0.0
lambda 11  = -9.875
lambda 12  = -0.0625
lambda 13  = 0.0
lambda 14  = 0.0
lambda 15  = 0.0
lambda 16  = 0.0
lambda 17  = 0.0
lambda 18  = 0.0
lambda 19  = 0.0
lambda 20  = 0.0
lambda 21  = 0.0
lambda 22  = 0.0
lambda 23  = 0.0
lambda 24  = 0.0
lambda 25  = 0.0
lambda 26  = 0.0
lambda 27  = 0.0
lambda 28  = 0.0
lambda 29  = 0.0
lambda 30  = 0.0
lambda 31  = 0.0
lambda 32  = 0.0
lambda 33  = 0.0
lambda 34  = 0.0
lambda 35  = 0.0
lambda 36  = 0.0
lambda 37  = 0.0
lambda 38  = 0.0
lambda 39  = 0.0
lambda 40  = 0.0
lambda 41  = 0.0
lambda 42  = 0.0
lambda 43  = 0.0
lambda 44  = 0.0
lambda 45  = 0.0
lambda 46  = 0.0
lambda 47  = 0.0
lambda 48  = 0.0
lambda 49  = 0.0
lambda 50  = 0.0
lambda 51  = 0.0
lambda 52  = 0.0
lambda 53  = 0.0
lambda 54  = 0.0
lambda 55  = 0.0
lambda 56  = 0.0
lambda 57  = 0.0
lambda 58  = 0.0
lambda 59  = 0.0
lambda 60  = 0.0
lambda 61  = 0.0
lambda 62  = 0.0
lambda 63  = 0.0
lambda 64  = 0.0
lambda 65  = 0.0
lambda 66  = 0.0
```

```
lambda 67  = 0.0
lambda 68  = 0.0
lambda 69  = 0.0
lambda 70  = 0.0
lambda 71  = 0.0
lambda 72  = 0.0
lambda 73  = 0.0
lambda 74  = 0.0
lambda 75  = 0.0
lambda 76  = 0.0
lambda 77  = 0.0
lambda 78  = 0.0
lambda 79  = 0.0
lambda 80  = 0.0
lambda 81  = 0.0
lambda 82  = 0.0
lambda 83  = 0.0
lambda 84  = 0.0
lambda 85  = 0.0
lambda 86  = 0.0
lambda 87  = 0.0
lambda 88  = 0.0
lambda 89  = 0.0
lambda 90  = 0.0
lambda 91  = 0.0
lambda 92  = 0.0
lambda 93  = 0.0
lambda 94  = 0.0
lambda 95  = 0.0
lambda 96  = 0.0
lambda 97  = 0.0
lambda 98  = 0.0
lambda 99  = 0.0
lambda 100  = 0.0
lambda 101  = 0.0
lambda 102  = 0.0
lambda 103  = 0.0
lambda 104  = 0.0
lambda 105  = 0.0
lambda 106  = 0.0
lambda 107  = 0.0
lambda 108  = 0.0
lambda 109  = 0.0
lambda 110  = 0.0
lambda 111  = 0.0
lambda 112  = 0.0
lambda 113  = 0.0
lambda 114  = 0.0
lambda 115  = 0.0
lambda 116  = 0.0
lambda 117  = 0.0
lambda 118  = 0.0
lambda 119  = 0.0
lambda 120  = 0.0
lambda 121  = 0.0
lambda 122  = 0.0
lambda 123  = 0.0
lambda 124  = 0.0
lambda 125  = 0.0
lambda 126  = 0.0
lambda 127  = 0.0
lambda 128  = 0.0
lambda 129  = 0.0
lambda 130  = 0.0
lambda 131  = 0.0
lambda 132  = 0.0
lambda 133  = 0.0
lambda 134  = 0.0
lambda 135  = 0.0
lambda 136  = 0.0
lambda 137  = 0.0
lambda 138  = 0.0
lambda 139  = 0.0
lambda 140  = 0.0
lambda 141  = 0.0
lambda 142  = 0.0
lambda 143  = 0.0
lambda 144  = 0.0
lambda 145  = 0.0
lambda 146  = 0.0
lambda 147  = 0.0
lambda 148  = 0.0
lambda 149  = 0.0
lambda 150  = 0.0
lambda 151  = 0.0
```

```
lambda 152  = 0.0
lambda 153  = 0.0
lambda 154  = 0.0
lambda 155  = 0.0
lambda 156  = 0.0
lambda 157  = 0.0
lambda 158  = 0.0
lambda 159  = 0.0
lambda 160  = 0.0
lambda 161  = 0.0
lambda 162  = 0.0
lambda 163  = 0.0
lambda 164  = 0.0
lambda 165  = 0.0
lambda 166  = 0.0
lambda 167  = 0.0
lambda 168  = 0.0
lambda 169  = 0.0
lambda 170  = 0.0
lambda 171  = 0.0
lambda 172  = 0.0
lambda 173  = 0.0
lambda 174  = 0.0
lambda 175  = 0.0
lambda 176  = 0.0
lambda 177  = 0.0
lambda 178  = 0.0
lambda 179  = 0.0
lambda 180  = 0.0
lambda 181  = 0.0
lambda 182  = 0.0
lambda 183  = 0.0
lambda 184  = 0.0
lambda 185  = 0.0
lambda 186  = 0.0
lambda 187  = 0.0
lambda 188  = 0.0
lambda 189  = 0.0
lambda 190  = 0.0
lambda 191  = 0.0
lambda 192  = 0.0
lambda 193  = 0.0
lambda 194  = 0.0
lambda 195  = 0.0
lambda 196  = 0.0
lambda 197  = 0.0
lambda 198  = 0.0
lambda 199  = 0.0
lambda 200  = 0.0
lambda 201  = 0.0
lambda 202  = 0.0
lambda 203  = 0.0
lambda 204  = 0.0
lambda 205  = 0.0
lambda 206  = 0.0
lambda 207  = 0.0
lambda 208  = 0.0
lambda 209  = 0.0
lambda 210  = 0.0
lambda 211  = 0.0
lambda 212  = 0.0
lambda 213  = 0.0
lambda 214  = 0.0
lambda 215  = 0.0
lambda 216  = 0.0
lambda 217  = 0.0
lambda 218  = 0.0
lambda 219  = 0.0
lambda 220  = 0.0
lambda 221  = 0.0
lambda 222  = 0.0
lambda 223  = 0.0
lambda 224  = 0.0
lambda 225  = 0.0
lambda 226  = 0.0
lambda 227  = 0.0
lambda 228  = 0.0
lambda 229  = 0.0
lambda 230  = 0.0
lambda 231  = 0.0
lambda 232  = 0.0
lambda 233  = 0.0
lambda 234  = 0.0
lambda 235  = 0.0
lambda 236  = 0.0
```

```
lambda 237  = 0.0
lambda 238  = 0.0
lambda 239  = 0.0
lambda 240  = 0.0
lambda 241  = 0.0
lambda 242  = 0.0
lambda 243  = 0.0
lambda 244  = 0.0
lambda 245  = 0.0
lambda 246  = 0.0
lambda 247  = 0.0
lambda 248  = 0.0
lambda 249  = 0.0
lambda 250  = 0.0
lambda 251  = 0.0
lambda 252  = 0.0
lambda 253  = 0.0
lambda 254  = 0.0
lambda 255  = 0.0
lambda 256  = 0.0
lambda 257  = 0.0
lambda 258  = 0.0
lambda 259  = 0.0
lambda 260  = 0.0
lambda 261  = 0.0
lambda 262  = 0.0
lambda 263  = 0.0
lambda 264  = 0.0
lambda 265  = 0.0
lambda 266  = 0.0
lambda 267  = 0.0
lambda 268  = 0.0
lambda 269  = 0.0
lambda 270  = 0.0
lambda 271  = 0.0
lambda 272  = 0.0
lambda 273  = 0.0
lambda 274  = 0.0
lambda 275  = 0.0
lambda 276  = 0.0
lambda 277  = 0.0
lambda 278  = 0.0
lambda 279  = 0.0
lambda 280  = 0.0
lambda 281  = 0.0
lambda 282  = 0.0
lambda 283  = 0.0
lambda 284  = 0.0
lambda 285  = 0.0
lambda 286  = 0.0
lambda 287  = 0.0
lambda 288  = 0.0
lambda 289  = 0.0
lambda 290  = 0.0
lambda 291  = 0.0
lambda 292  = 0.0
lambda 293  = 0.0
lambda 294  = 0.0
lambda 295  = 0.0
lambda 296  = 0.0
lambda 297  = 0.0
lambda 298  = 0.0
lambda 299  = 0.0
lambda 300  = 0.0
lambda 301  = 0.0
lambda 302  = 0.0
lambda 303  = 0.0
lambda 304  = 0.0
lambda 305  = 0.0
lambda 306  = 0.0
lambda 307  = 0.0
lambda 308  = 0.0
lambda 309  = 0.0
lambda 310  = 0.0
lambda 311  = 0.0
lambda 312  = 0.0
lambda 313  = 0.0
lambda 314  = 0.0
lambda 315  = 0.0
lambda 316  = 0.0
lambda 317  = 0.0
lambda 318  = 0.0
lambda 319  = 0.0
lambda 320  = 0.0
lambda 321  = 0.0
```

```
lambda 322  = 0.0
lambda 323  = 0.0
lambda 324  = 0.0
lambda 325  = 0.0
lambda 326  = 0.0
lambda 327  = 0.0
lambda 328  = 0.0
lambda 329  = 0.0
lambda 330  = 0.0
lambda 331  = 0.0
lambda 332  = 0.0
lambda 333  = 0.0
lambda 334  = 0.0
lambda 335  = 0.0
lambda 336  = 0.0
lambda 337  = 0.0
lambda 338  = 0.0
lambda 339  = 0.0
lambda 340  = 0.0
lambda 341  = 0.0
lambda 342  = 0.0
lambda 343  = 0.0
lambda 344  = 0.0
lambda 345  = 0.0
lambda 346  = 0.0
lambda 347  = 0.0
lambda 348  = 0.0
lambda 349  = 0.0
lambda 350  = 0.0
lambda 351  = 0.0
lambda 352  = 0.0
lambda 353  = 0.0
lambda 354  = 0.0
lambda 355  = 0.0
lambda 356  = 0.0
lambda 357  = 0.0
lambda 358  = 0.0
lambda 359  = 0.0
lambda 360  = 0.0
lambda 361  = 0.0
lambda 362  = 0.0
lambda 363  = 0.0
lambda 364  = 0.0
lambda 365  = 0.0
lambda 366  = 0.0
lambda 367  = 0.0
lambda 368  = 0.0
lambda 369  = 0.0
lambda 370  = 0.0
lambda 371  = 0.0
lambda 372  = 0.0
lambda 373  = 0.0
lambda 374  = 0.0
lambda 375  = 0.0
lambda 376  = 0.0
lambda 377  = 0.0
lambda 378  = 0.0
lambda 379  = 0.0
lambda 380  = 0.0
lambda 381  = 0.0
lambda 382  = 0.0
lambda 383  = 0.0
lambda 384  = 0.0
lambda 385  = 0.0
lambda 386  = 0.0
lambda 387  = 0.0
lambda 388  = 0.0
lambda 389  = 0.0
lambda 390  = 0.0
lambda 391  = 0.0
lambda 392  = 0.0
lambda 393  = 0.0
lambda 394  = 0.0
lambda 395  = 0.0
lambda 396  = 0.0
lambda 397  = 0.0
lambda 398  = 0.0
lambda 399  = 0.0
lambda 400  = 0.0
lambda 401  = 0.0
lambda 402  = 0.0
lambda 403  = 0.0
lambda 404  = 0.0
lambda 405  = 0.0
lambda 406  = 0.0
```

```
lambda 407  = 0.0
lambda 408  = 0.0
lambda 409  = 0.0
lambda 410  = 0.0
lambda 411  = 0.0
lambda 412  = 0.0
lambda 413  = 0.0
lambda 414  = 0.0
lambda 415  = 0.0
lambda 416  = 0.0
lambda 417  = 0.0
lambda 418  = 0.0
lambda 419  = 0.0
lambda 420  = 0.0
lambda 421  = 0.0
lambda 422  = 0.0
lambda 423  = 0.0
lambda 424  = 0.0
lambda 425  = 0.0
lambda 426  = 0.0
lambda 427  = 0.0
lambda 428  = 0.0
lambda 429  = 0.0
lambda 430  = 0.0
lambda 431  = 0.0
lambda 432  = 0.0
lambda 433  = 0.0
lambda 434  = 0.0
lambda 435  = 0.0
lambda 436  = 0.0
lambda 437  = 0.0
lambda 438  = 0.0
lambda 439  = 0.0
lambda 440  = 0.0
lambda 441  = 0.0
lambda 442  = 0.0
lambda 443  = 0.0
lambda 444  = 0.0
lambda 445  = 0.0
lambda 446  = 0.0
lambda 447  = 0.0
lambda 448  = 0.0
lambda 449  = 0.0
lambda 450  = 0.0
lambda 451  = 0.0
lambda 452  = 0.0
lambda 453  = 0.0
lambda 454  = 0.0
lambda 455  = 0.0
lambda 456  = 0.0
lambda 457  = 0.0
lambda 458  = 0.0
lambda 459  = 0.0
lambda 460  = 0.0
lambda 461  = 0.0
lambda 462  = 0.0
lambda 463  = 0.0
lambda 464  = 0.0
lambda 465  = 0.0
lambda 466  = 0.0
lambda 467  = 0.0
lambda 468  = 0.0
lambda 469  = 0.0
lambda 470  = 0.0
lambda 471  = 0.0
lambda 472  = 0.0
lambda 473  = 0.0
lambda 474  = 0.0
lambda 475  = 0.0
lambda 476  = 0.0
lambda 477  = 0.0
lambda 478  = 0.0
lambda 479  = 0.0
lambda 480  = 0.0
lambda 481  = 0.0
lambda 482  = 0.0
lambda 483  = 0.0
lambda 484  = 0.0
lambda 485  = 0.0
lambda 486  = 0.0
lambda 487  = 0.0
lambda 488  = 0.0
lambda 489  = 0.0
lambda 490  = 0.0
lambda 491  = 0.0
```

```
lambda 492  = 0.0
lambda 493  = 0.0
lambda 494  = 0.0
lambda 495  = 0.0
lambda 496  = 0.0
lambda 497  = 0.0
lambda 498  = 0.0
lambda 499  = 0.0
lambda 500  = 0.0
lambda 501  = 0.0
lambda 502  = 0.0
lambda 503  = 0.0
lambda 504  = 0.0
lambda 505  = 0.0
lambda 506  = 0.0
lambda 507  = 0.0
lambda 508  = 0.0
lambda 509  = 0.0
lambda 510  = 0.0
lambda 511  = 0.0
lambda 512  = 0.0
lambda 513  = 0.0
lambda 514  = 0.0
lambda 515  = 0.0
lambda 516  = 0.0
lambda 517  = 0.0
lambda 518  = 0.0
lambda 519  = 0.0
lambda 520  = 0.0
lambda 521  = 0.0
lambda 522  = 0.0
lambda 523  = 0.0
lambda 524  = 0.0
lambda 525  = 0.0
lambda 526  = 0.0
lambda 527  = 0.0
lambda 528  = 0.0
lambda 529  = 0.0
lambda 530  = 0.0
lambda 531  = 0.0
lambda 532  = 0.0
lambda 533  = 0.0
lambda 534  = 0.0
lambda 535  = 0.0
lambda 536  = 0.0
lambda 537  = 0.0
lambda 538  = 0.0
lambda 539  = 0.0
lambda 540  = 0.0
lambda 541  = 0.0
lambda 542  = 0.0
lambda 543  = 0.0
lambda 544  = 0.0
lambda 545  = 0.0
lambda 546  = 0.0
lambda 547  = 0.0
lambda 548  = 0.0
lambda 549  = 0.0
lambda 550  = 0.0
lambda 551  = 0.0
lambda 552  = 0.0
lambda 553  = 0.0
lambda 554  = 0.0
lambda 555  = 0.0
lambda 556  = 0.0
lambda 557  = 0.0
lambda 558  = 0.0
lambda 559  = 0.0
lambda 560  = 0.0
lambda 561  = 0.0
lambda 562  = 0.0
lambda 563  = 0.0
lambda 564  = 0.0
lambda 565  = 0.0
lambda 566  = 0.0
lambda 567  = 0.0
lambda 568  = 0.0
lambda 569  = 0.0
lambda 570  = 0.0
lambda 571  = 0.0
lambda 572  = 0.0
lambda 573  = 0.0
lambda 574  = 0.0
lambda 575  = 0.0
lambda 576  = 0.0
```

```
lambda 577  = 0.0
lambda 578  = 0.0
lambda 579  = 0.0
lambda 580  = 0.0
lambda 581  = 0.0
lambda 582  = 0.0
lambda 583  = 0.0
lambda 584  = 0.0
lambda 585  = 0.0
lambda 586  = 0.0
lambda 587  = 0.0
lambda 588  = 0.0
lambda 589  = 0.0
lambda 590  = 0.0
lambda 591  = 0.0
lambda 592  = 0.0
lambda 593  = 0.0
lambda 594  = 0.0
lambda 595  = 0.0
lambda 596  = 0.0
lambda 597  = 0.0
lambda 598  = 0.0
lambda 599  = 0.0
lambda 600  = 0.0
lambda 601  = 0.0
lambda 602  = 0.0
lambda 603  = 0.0
lambda 604  = 0.0
lambda 605  = 0.0
lambda 606  = 0.0
lambda 607  = 0.0
lambda 608  = 0.0
lambda 609  = 0.0
lambda 610  = 0.0
lambda 611  = 0.0
lambda 612  = 0.0
lambda 613  = 0.0
lambda 614  = 0.0
lambda 615  = 0.0
lambda 616  = 0.0
lambda 617  = 0.0
lambda 618  = 0.0
lambda 619  = 0.0
lambda 620  = 0.0
lambda 621  = 0.0
lambda 622  = 0.0
lambda 623  = 0.0
lambda 624  = 0.0
lambda 625  = 0.0
lambda 626  = 0.0
lambda 627  = 0.0
lambda 628  = 0.0
lambda 629  = 0.0
lambda 630  = 0.0
lambda 631  = 0.0
lambda 632  = 0.0
lambda 633  = 0.0
lambda 634  = 0.0
lambda 635  = 0.0
lambda 636  = 0.0
lambda 637  = 0.0
lambda 638  = 0.0
lambda 639  = 0.0
lambda 640  = 0.0
lambda 641  = 0.0
lambda 642  = 0.0
lambda 643  = 0.0
lambda 644  = 0.0
lambda 645  = 0.0
lambda 646  = 0.0
lambda 647  = 0.0
lambda 648  = 0.0
lambda 649  = 0.0
lambda 650  = 0.0
lambda 651  = 0.0
lambda 652  = 0.0
lambda 653  = 0.0
lambda 654  = 0.0
lambda 655  = 0.0
lambda 656  = 0.0
lambda 657  = 0.0
lambda 658  = 0.0
lambda 659  = 0.0
lambda 660  = 0.0
lambda 661  = 0.0
```

```
lambda 662  = 0.0
lambda 663  = 0.0
lambda 664  = 0.0
lambda 665  = 0.0
lambda 666  = 0.0
lambda 667  = 0.0
lambda 668  = 0.0
lambda 669  = 0.0
lambda 670  = 0.0
lambda 671  = 0.0
lambda 672  = 0.0
lambda 673  = 0.0
lambda 674  = 0.0
lambda 675  = 0.0
lambda 676  = 0.0
lambda 677  = 0.0
lambda 678  = 0.0
lambda 679  = 0.0
lambda 680  = 0.0
lambda 681  = 0.0
lambda 682  = 0.0
lambda 683  = 0.0
lambda 684  = 0.0
lambda 685  = 0.0
lambda 686  = 0.0
lambda 687  = 0.0
lambda 688  = 0.0
lambda 689  = 0.0
lambda 690  = 0.0
lambda 691  = 0.0
lambda 692  = 0.0
lambda 693  = 0.0
lambda 694  = 0.0
lambda 695  = 0.0
lambda 696  = 0.0
lambda 697  = 0.0
lambda 698  = 0.0
lambda 699  = 0.0
lambda 700  = 0.0
lambda 701  = 0.0
lambda 702  = 0.0
lambda 703  = 0.0
lambda 704  = 0.0
lambda 705  = 0.0
lambda 706  = 0.0
lambda 707  = 0.0
lambda 708  = 0.0
lambda 709  = 0.0
lambda 710  = 0.0
lambda 711  = 0.0
lambda 712  = 0.0
lambda 713  = 0.0
lambda 714  = 0.0
lambda 715  = 0.0
lambda 716  = 0.0
lambda 717  = 0.0
lambda 718  = 0.0
lambda 719  = 0.0
lambda 720  = 0.0
lambda 721  = 0.0
lambda 722  = 0.0
lambda 723  = 0.0
lambda 724  = 0.0
lambda 725  = 0.0
lambda 726  = 0.0
lambda 727  = 0.0
lambda 728  = 0.0
lambda 729  = 0.0
lambda 730  = 0.0
lambda 731  = 0.0
lambda 732  = 0.0
lambda 733  = 0.0
lambda 734  = 0.0
lambda 735  = 0.0
lambda 736  = 0.0
lambda 737  = 0.0
lambda 738  = 0.0
lambda 739  = 0.0
lambda 740  = 0.0
lambda 741  = 0.0
lambda 742  = 0.0
lambda 743  = 0.0
lambda 744  = 0.0
lambda 745  = 0.0
lambda 746  = 0.0
```

```
lambda 747  = 0.0
lambda 748  = 0.0
lambda 749  = 0.0
lambda 750  = 0.0
lambda 751  = 0.0
lambda 752  = 0.0
lambda 753  = 0.0
lambda 754  = 0.0
lambda 755  = 0.0
lambda 756  = 0.0
lambda 757  = 0.0
lambda 758  = 0.0
lambda 759  = 0.0
lambda 760  = 0.0
lambda 761  = 0.0
lambda 762  = 0.0
lambda 763  = 0.0
lambda 764  = 0.0
lambda 765  = 0.0
lambda 766  = 0.0
lambda 767  = 0.0
lambda 768  = 0.0
lambda 769  = 0.0
lambda 770  = 0.0
lambda 771  = 0.0
lambda 772  = 0.0
lambda 773  = 0.0
lambda 774  = 0.0
lambda 775  = 0.0
lambda 776  = 0.0
lambda 777  = 0.0
lambda 778  = 0.0
lambda 779  = 0.0
lambda 780  = 0.0
lambda 781  = 0.0
lambda 782  = 0.0
lambda 783  = 0.0
lambda 784  = 0.0
lambda 785  = 0.0
lambda 786  = 0.0
lambda 787  = 0.0
lambda 788  = 0.0
lambda 789  = 0.0
lambda 790  = 0.0
lambda 791  = 0.0
lambda 792  = 0.0
lambda 793  = 0.0
lambda 794  = 0.0
lambda 795  = 0.0
lambda 796  = 0.0
lambda 797  = 0.0
lambda 798  = 0.0
lambda 799  = 0.0
lambda 800  = 0.0
lambda 801  = 0.0
lambda 802  = 0.0
lambda 803  = 0.0
lambda 804  = 0.0
lambda 805  = 0.0
lambda 806  = 0.0
lambda 807  = 0.0
lambda 808  = 0.0
lambda 809  = 0.0
lambda 810  = 0.0
lambda 811  = 0.0
lambda 812  = 0.0
lambda 813  = 0.0
lambda 814  = 0.0
lambda 815  = 0.0
lambda 816  = 0.0
lambda 817  = 0.0
lambda 818  = 0.0
lambda 819  = 0.0
lambda 820  = 0.0
lambda 821  = 0.0
lambda 822  = 0.0
lambda 823  = 0.0
lambda 824  = 0.0
lambda 825  = 0.0
lambda 826  = 0.0
lambda 827  = 0.0
lambda 828  = 0.0
lambda 829  = 0.0
lambda 830  = 0.0
lambda 831  = 0.0
```

```
lambda 832  = 0.0
lambda 833  = 0.0
lambda 834  = 0.0
lambda 835  = 0.0
lambda 836  = 0.0
lambda 837  = 0.0
lambda 838  = 0.0
lambda 839  = 0.0
lambda 840  = 0.0
lambda 841  = 0.0
lambda 842  = 0.0
lambda 843  = 0.0
lambda 844  = 0.0
lambda 845  = 0.0
lambda 846  = 0.0
lambda 847  = 0.0
lambda 848  = 0.0
lambda 849  = 0.0
lambda 850  = 0.0
lambda 851  = 0.0
lambda 852  = 0.0
lambda 853  = 0.0
lambda 854  = 0.0
lambda 855  = 0.0
lambda 856  = 0.0
lambda 857  = 0.0
lambda 858  = 0.0
lambda 859  = 0.0
lambda 860  = 0.0
lambda 861  = 0.0
lambda 862  = 0.0
lambda 863  = 0.0
lambda 864  = 0.0
lambda 865  = 0.0
lambda 866  = 0.0
lambda 867  = 0.0
lambda 868  = 0.0
lambda 869  = 0.0
lambda 870  = 0.0
lambda 871  = 0.0
lambda 872  = 0.0
lambda 873  = 0.0
lambda 874  = 0.0
lambda 875  = 0.0
lambda 876  = 0.0
lambda 877  = 0.0
lambda 878  = 0.0
lambda 879  = 0.0
lambda 880  = 0.0
lambda 881  = 0.0
lambda 882  = 0.0
lambda 883  = 0.0
lambda 884  = 0.0
lambda 885  = 0.0
lambda 886  = 0.0
lambda 887  = 0.0
lambda 888  = 0.0
lambda 889  = 0.0
lambda 890  = 0.0
lambda 891  = 0.0
lambda 892  = 0.0
lambda 893  = 0.0
lambda 894  = 0.0
lambda 895  = 0.0
lambda 896  = 0.0
lambda 897  = 0.0
lambda 898  = 0.0
lambda 899  = 0.0
lambda 900  = 0.0
lambda 901  = 0.0
lambda 902  = 0.0
lambda 903  = 0.0
lambda 904  = 0.0
lambda 905  = 0.0
lambda 906  = 0.0
lambda 907  = 0.0
lambda 908  = 0.0
lambda 909  = 0.0
lambda 910  = 0.0
lambda 911  = 0.0
lambda 912  = 0.0
lambda 913  = 0.0
lambda 914  = 0.0
lambda 915  = 0.0
lambda 916  = 0.0
```

```
lambda 917  = 0.0
lambda 918  = 0.0
lambda 919  = 0.0
lambda 920  = 0.0
lambda 921  = 0.0
lambda 922  = 0.0
lambda 923  = 0.0
lambda 924  = 0.0
lambda 925  = 0.0
lambda 926  = 0.0
lambda 927  = 0.0
lambda 928  = 0.0
lambda 929  = 0.0
lambda 930  = 0.0
lambda 931  = 0.0
lambda 932  = 0.0
lambda 933  = 0.0
lambda 934  = 0.0
lambda 935  = 0.0
lambda 936  = 0.0
lambda 937  = 0.0
lambda 938  = 0.0
lambda 939  = 0.0
lambda 940  = 0.0
lambda 941  = 0.0
lambda 942  = 0.0
lambda 943  = 0.0
lambda 944  = 0.0
lambda 945  = 0.0
lambda 946  = 0.0
lambda 947  = 0.0
lambda 948  = 0.0
lambda 949  = 0.0
lambda 950  = 0.0
lambda 951  = 0.0
lambda 952  = 0.0
lambda 953  = 0.0
lambda 954  = 0.0
lambda 955  = 0.0
lambda 956  = 0.0
lambda 957  = 0.0
lambda 958  = 0.0
lambda 959  = 0.0
lambda 960  = 0.0
lambda 961  = 0.0
lambda 962  = 0.0
lambda 963  = 0.0
lambda 964  = 0.0
lambda 965  = 0.0
lambda 966  = 0.0
lambda 967  = 0.0
lambda 968  = 0.0
lambda 969  = 0.0
lambda 970  = 0.0
lambda 971  = 0.0
lambda 972  = 0.0
lambda 973  = 0.0
lambda 974  = 0.0
lambda 975  = 0.0
lambda 976  = 0.0
lambda 977  = 0.0
lambda 978  = 0.0
lambda 979  = 0.0
lambda 980  = 0.0
lambda 981  = 0.0
lambda 982  = 0.0
lambda 983  = 0.0
lambda 984  = 0.0
lambda 985  = 0.0
lambda 986  = 0.0
lambda 987  = 0.0
lambda 988  = 0.0
lambda 989  = 0.0
lambda 990  = 0.0
lambda 991  = 0.0
lambda 992  = 0.0
lambda 993  = 0.0
lambda 994  = 0.0
lambda 995  = 0.0
lambda 996  = 0.0
lambda 997  = 0.0
lambda 998  = 0.0
lambda 999  = 0.0
lambda 1000  = 0.0
lambda 1001  = 0.0
```

```
lambda 1002  = 0.0
lambda 1003  = 0.0
lambda 1004  = 0.0
lambda 1005  = 0.0
lambda 1006  = 0.0
lambda 1007  = 0.0
lambda 1008  = 0.0
lambda 1009  = 0.0
lambda 1010  = 0.0
lambda 1011  = 0.0
lambda 1012  = 0.0
lambda 1013  = 0.0
lambda 1014  = 0.0
lambda 1015  = 0.0
lambda 1016  = 0.0
lambda 1017  = 0.0
lambda 1018  = 0.0
lambda 1019  = 0.0
lambda 1020  = 0.0
lambda 1021  = 0.0
lambda 1022  = 0.0
lambda 1023  = 0.0
lambda 1024  = 0.0
lambda 1025  = 0.0
lambda 1026  = 0.0
lambda 1027  = 0.0
lambda 1028  = 0.0
lambda 1029  = 0.0
lambda 1030  = 0.0
lambda 1031  = 0.0
lambda 1032  = 0.0
lambda 1033  = 0.0
lambda 1034  = 0.0
lambda 1035  = 0.0
lambda 1036  = 0.0
lambda 1037  = 0.0
lambda 1038  = 0.0

Variables solutions below...
X0_0                    0.0
X0_1                    0.0
X0_2                    1.0
X1_0                    0.0
X1_1                    0.0
X1_2                    0.0
X2_0                    1.0
X2_1                    0.0
X2_2                    0.0
X3_0                    0.0
X3_1                    0.0
X3_2                    0.0
X4_0                    0.0
X4_1                    1.0
X4_2                    0.0
X5_0                    0.0
X5_1                    0.0
X5_2                    0.0
U0                      1.0
U1                      0.0
U2                      1.0
U3                      0.0
U4                      1.0
U5                      0.0
Z16_0                   0.0
Z8_0                    0.0
Z9_0                    0.0
Z10_0                   37.0
Z11_0                   100.0
Z16_1                   0.0
Z8_1                    0.0
Z9_1                    0.0
Z10_1                   0.0
Z11_1                   0.0
Z16_2                   200.0
Z8_2                    0.0
Z9_2                    0.0
Z10_2                   0.0
Z11_2                   0.0
Z0_0                    0.0
Z1_0                    0.0
Z17_0                   0.0
Z0_1                    0.0
Z1_1                    0.0
Z17_1                   0.0
Z0_2                    0.0
```

Z1_2	0.0
Z17_2	0.0
Z2_0	0.0
Z3_0	0.0
Z18_0	0.0
Z2_1	0.0
Z3_1	0.0
Z18_1	0.0
Z2_2	63.0
Z3_2	0.0
Z18_2	0.0
Z19_0	0.0
Z4_0	0.0
Z5_0	0.0
Z19_1	0.0
Z4_1	0.0
Z5_1	0.0
Z19_2	0.0
Z4_2	0.0
Z5_2	0.0
Z20_0	0.0
Z6_0	0.0
Z7_0	0.0
Z20_1	0.0
Z6_1	0.0
Z7_1	0.0
Z20_2	0.0
Z6_2	0.0
Z7_2	0.0
Z12_0	0.0
Z13_0	0.0
Z14_0	0.0
Z15_0	0.0
Z21_0	0.0
Z12_1	0.0
Z13_1	0.0
Z14_1	100.0
Z15_1	100.0
Z21_1	0.0
Z12_2	0.0
Z13_2	0.0
Z14_2	0.0
Z15_2	0.0
Z21_2	0.0
Y0_0	0.0
Y0_1	0.0
Y0_2	0.0
Y1_0	0.0
Y1_1	0.0
Y1_2	0.0
Y2_0	0.0
Y2_1	0.0
Y2_2	1.0
Y3_0	0.0
Y3_1	0.0
Y3_2	0.0
Y4_0	0.0
Y4_1	0.0
Y4_2	0.0
Y5_0	0.0
Y5_1	0.0
Y5_2	0.0
Y6_0	0.0
Y6_1	0.0
Y6_2	0.0
Y7_0	0.0
Y7_1	0.0
Y7_2	0.0
Y8_0	0.0
Y8_1	0.0
Y8_2	0.0
Y9_0	0.0
Y9_1	0.0
Y9_2	0.0
Y10_0	1.0
Y10_1	0.0
Y10_2	0.0
Y11_0	1.0
Y11_1	0.0
Y11_2	0.0
Y12_0	0.0
Y12_1	0.0
Y12_2	0.0
Y13_0	0.0
Y13_1	0.0

Y13_2	0.0
Y14_0	0.0
Y14_1	1.0
Y14_2	0.0
Y15_0	0.0
Y15_1	1.0
Y15_2	0.0
Y16_0	0.0
Y16_1	0.0
Y16_2	1.0
Y17_0	0.0
Y17_1	0.0
Y17_2	0.0
Y18_0	0.0
Y18_1	0.0
Y18_2	0.0
Y19_0	0.0
Y19_1	0.0
Y19_2	0.0
Y20_0	0.0
Y20_1	0.0
Y20_2	0.0
Y21_0	0.0
Y21_1	0.0
Y21_2	0.0
V0	100.0
V1	100.0
V2	100.0
V3	0.0
V4	0.0
V5	100.0
V6	100.0
V7	100.0
V8	200.0
V9	0.0
V10	0.0
V11	0.0
V12	0.0
V13	0.0
a0	2.0
a1	2.0
a2	2.0
a3	0.0
a4	0.0
a5	2.0
a6	2.0
a7	2.0
a8	4.0
a9	0.0
a10	0.0
a11	0.0
a12	0.0
a13	0.0
c0_0	1.0
c0_1	1.0
c0_2	0.0
c1_0	1.0
c1_1	1.0
c1_2	0.0
c2_0	1.0
c2_1	1.0
c2_2	0.0
c3_0	0.0
c3_1	1.0
c3_2	1.0
c4_0	0.0
c4_1	1.0
c4_2	1.0
c5_0	1.0
c5_1	1.0
c5_2	0.0
c6_0	1.0
c6_1	1.0
c6_2	0.0
c7_0	1.0
c7_1	1.0
c7_2	0.0
c8_0	1.0
c8_1	1.0
c8_2	1.0
c8_3	1.0
c8_4	0.0
c8_5	1.0
c8_6	1.0
c8_7	1.0

c8_8	1.0
c8_9	1.0
c8_10	1.0
c8_11	1.0
c8_12	1.0
c8_13	1.0
c8_14	1.0
c8_15	1.0
c8_16	1.0
c8_17	1.0
c8_18	1.0
c8_19	1.0
c8_20	1.0
c9_0	0.0
c9_1	1.0
c9_2	1.0
c9_3	1.0
c9_4	1.0
c9_5	1.0
c9_6	1.0
c9_7	1.0
c9_8	1.0
c9_9	1.0
c9_10	1.0
c9_11	1.0
c9_12	1.0
c9_13	1.0
c9_14	1.0
c9_15	1.0
c9_16	1.0
c9_17	1.0
c9_18	1.0
c9_19	1.0
c9_20	1.0
c10_0	0.0
c10_1	1.0
c10_2	1.0
c10_3	1.0
c10_4	1.0
c10_5	1.0
c10_6	1.0
c10_7	1.0
c10_8	1.0
c10_9	1.0
c10_10	1.0
c10_11	1.0
c10_12	1.0
c10_13	1.0
c10_14	1.0
c10_15	1.0
c10_16	1.0
c10_17	1.0
c10_18	1.0
c10_19	1.0
c10_20	1.0
c11_0	0.0
c11_1	1.0
c11_2	1.0
c11_3	1.0
c11_4	1.0
c11_5	1.0
c11_6	1.0
c11_7	1.0
c11_8	1.0
c11_9	1.0
c11_10	1.0
c11_11	1.0
c11_12	1.0
c11_13	1.0
c11_14	1.0
c11_15	1.0
c11_16	1.0
c11_17	1.0
c11_18	1.0
c11_19	1.0
c11_20	1.0
c12_0	0.0
c12_1	1.0
c12_2	1.0
c12_3	1.0
c12_4	1.0
c12_5	1.0
c12_6	1.0
c12_7	1.0
c12_8	1.0

c12_9	1.0
c12_10	1.0
c12_11	1.0
c12_12	1.0
c12_13	1.0
c12_14	1.0
c12_15	1.0
c12_16	1.0
c12_17	1.0
c12_18	1.0
c12_19	1.0
c12_20	1.0
c13_0	0.0
c13_1	1.0
c13_2	1.0
c13_3	1.0
c13_4	1.0
c13_5	1.0
c13_6	1.0
c13_7	1.0
c13_8	1.0
c13_9	1.0
c13_10	1.0
c13_11	1.0
c13_12	1.0
c13_13	1.0
c13_14	1.0
c13_15	1.0
c13_16	1.0
c13_17	1.0
c13_18	1.0
c13_19	1.0
c13_20	1.0
k0_0	0.0
k0_1	0.0
k0_2	1.0
k1_0	0.0
k1_1	0.0
k1_2	1.0
k2_0	0.0
k2_1	0.0
k2_2	1.0
k3_0	1.0
k3_1	0.0
k3_2	0.0
k4_0	1.0
k4_1	0.0
k4_2	0.0
k5_0	0.0
k5_1	0.0
k5_2	1.0
k6_0	0.0
k6_1	0.0
k6_2	1.0
k7_0	0.0
k7_1	0.0
k7_2	1.0
k8_0	0.0
k8_1	0.0
k8_2	0.0
k8_3	0.0
k8_4	1.0
k8_5	0.0
k8_6	0.0
k8_7	0.0
k8_8	0.0
k8_9	0.0
k8_10	0.0
k8_11	0.0
k8_12	0.0
k8_13	0.0
k8_14	0.0
k8_15	0.0
k8_16	0.0
k8_17	0.0
k8_18	0.0
k8_19	0.0
k8_20	0.0
k9_0	1.0
k9_1	0.0
k9_2	0.0
k9_3	0.0
k9_4	0.0
k9_5	0.0
k9_6	0.0

k9_7	0.0
k9_8	0.0
k9_9	0.0
k9_10	0.0
k9_11	0.0
k9_12	0.0
k9_13	0.0
k9_14	0.0
k9_15	0.0
k9_16	0.0
k9_17	0.0
k9_18	0.0
k9_19	0.0
k9_20	0.0
k10_0	1.0
k10_1	0.0
k10_2	0.0
k10_3	0.0
k10_4	0.0
k10_5	0.0
k10_6	0.0
k10_7	0.0
k10_8	0.0
k10_9	0.0
k10_10	0.0
k10_11	0.0
k10_12	0.0
k10_13	0.0
k10_14	0.0
k10_15	0.0
k10_16	0.0
k10_17	0.0
k10_18	0.0
k10_19	0.0
k10_20	0.0
k11_0	1.0
k11_1	0.0
k11_2	0.0
k11_3	0.0
k11_4	0.0
k11_5	0.0
k11_6	0.0
k11_7	0.0
k11_8	0.0
k11_9	0.0
k11_10	0.0
k11_11	0.0
k11_12	0.0
k11_13	0.0
k11_14	0.0
k11_15	0.0
k11_16	0.0
k11_17	0.0
k11_18	0.0
k11_19	0.0
k11_20	0.0
k12_0	1.0
k12_1	0.0
k12_2	0.0
k12_3	0.0
k12_4	0.0
k12_5	0.0
k12_6	0.0
k12_7	0.0
k12_8	0.0
k12_9	0.0
k12_10	0.0
k12_11	0.0
k12_12	0.0
k12_13	0.0
k12_14	0.0
k12_15	0.0
k12_16	0.0
k12_17	0.0
k12_18	0.0
k12_19	0.0
k12_20	0.0
k13_0	1.0
k13_1	0.0
k13_2	0.0
k13_3	0.0
k13_4	0.0
k13_5	0.0
k13_6	0.0
k13_7	0.0

k13_8	0.0
k13_9	0.0
k13_10	0.0
k13_11	0.0
k13_12	0.0
k13_13	0.0
k13_14	0.0
k13_15	0.0
k13_16	0.0
k13_17	0.0
k13_18	0.0
k13_19	0.0
k13_20	0.0
G0	500.0
G1	500.0
G2	500.0
G3	0.0
G4	0.0
G5	500.0
G6	500.0
G7	500.0
G8	12.0
G9	0.0
G10	0.0
G11	0.0
G12	0.0
G13	0.0
H0	500.0
H1	1000.0
H2	500.0
H3	1000.0
H4	500.0
H5	1000.0
H6	500.0
H7	1000.0
H8	500.0
H9	1000.0
H10	500.0
H11	1000.0
H12	500.0
H13	1000.0
H14	500.0
H15	1000.0
H16	12.0
H17	0.0
H18	0.0
H19	0.0
H20	0.0
H21	0.0
r0_0_0_16	1.0
r0_0_2_16	0.0
r0_2_2_2	0.0
r0_2_2_3	1.0
r1_0_0_8	1.0
r1_0_0_9	1.0
r1_0_2_8	1.0
r1_0_2_9	1.0
r2_0_0_10	0.0
r2_0_0_11	0.0
r2_0_2_10	1.0
r2_0_2_11	1.0
r2_2_2_18	1.0
r3_5_1_12	1.0
r3_5_1_13	1.0
r4_5_1_14	0.0
r4_5_1_15	0.0
r5_5_1_21	1.0
