Archive Planning for a Distributed PACS

by

Peng Zhou

A Thesis submitted to the Faculty of Graduate Studies of

The University of Manitoba

in partial fulfilment of the requirements of the degree of

MASTER OF SCIENCE

Department of Computer Science

The University of Manitoba

Winnipeg, Manitoba

April 2008

THE UNIVERSITY OF MANITOBA

FACULTY OF GRADUATE STUDIES

*****

COPYRIGHT PERMISSION

Archive Planning for a Distributed PACS

BY

Peng Zhou

A Thesis/Practicum submitted to the Faculty of Graduate Studies of The University of Manitoba in partial fulfillment of the requirement of the degree

MASTER OF SCIENCE

Thesis advisors

Yanni Ellen Liu, Michel Toulouse

Author

Peng Zhou

Abstract

In the health industry, digital imaging data are stored and processed by a system called Picture Archive and Communication System (PACS). There are two types of architectures that can be chosen to implement a PACS: centralized and distributed. The centralized architecture has only one archive for storing and processing the digital imaging data. While the centralized architecture is currently the most common architecture, it does not provide the required scalability, reliability, and performance when the amount of digital medical images grows and the mobility of the image users increases. A distributed architecture of PACS's has more than one archive to share the load of storing and processing the digital medical image data. The database is separated physically but integrated logically [1]. The distributed architecture can overcome the limitations of centralized architecture and has advantages of reduced bottleneck, higher reliability and availability, higher modularity and thus higher scalability [1]. To design a distributed PACS, we need to determine the number, the location, and the capacity of the PACS archives. We also need to determine where to store each image file so that we can maximize the archive accessing performance within a certain budget. This thesis provides a methodology that helps PACS administrators to design a distributed PACS. This methodology can also be helpful to analyze the performance of an existing distributed PACS.

Table of Contents

List of Figures

List of Tables

Acknowledgments

I would like to begin by thanking my advisors, Dr. Yanni Ellen Liu and Dr. Michel Toulouse, for their guidance and support.

I am very grateful for Dr. Jeff Diamond's creative ideas on solving some critical issues in my thesis project.

As always, I am thankful to my parents who have supported me along the way.

Finally, I would like to acknowledge TRLabs, Winnipeg for providing partial financial support and research facilities to help me pursue my master's degree.

Chapter 1 Introduction

In the health industry, medical image acquisition equipments called modalities, e.g., computed tomography (CT), magnetic resonance imaging (MRI), ultrasound (US), nuclear medicine (NM), and computed radiography (CR), are used in the radiology department to do diagnostic imaging. Modern technologies have enhanced those imaging devices with capabilities to produce digital images. Both the number and the size of the digital images produced by those devices are very large. For example, currently a single Cardiology study using digital modalities produces 5-10 of Gigabytes of data [2] . Typically, a moderate size hospital can produce around 10 Terabytes of images per year [3]. Management information, radiologists' dictation, specialists' consultation and other additional information related to those diagnostic images add more data. To manage this large amount of data, a special system called Picture Archive and Communication System (PACS) was introduced in mid-1980s [4] for use in the radiology department. PACS collects the image data from modalities, stores the image data and related information created in the radiology work flow in archives, processes the data according to various requirements from clinicians and radiologists, and distributes the data to the proper users.

There are two types of architectures that can be chosen to build a PACS: centralized and distributed [5]. The centralized architecture has only one archive for storing and processing the digital medical data. The centralized architecture is easy to design, implement and manage but the malfunction of the central archive will bring down the entire PACS. Moreover, expanding the capacity of the single archive is limited by storage

hardware, network bandwidth, network coverage, and other factors. In the distributed architecture, on the other hand, there is more than one archive to share the load of storing and processing the digital medical data. Distribution of the data increases the reliability of the system. It can also be easier to add additional archives to improve the system performance when there are more users and medical imaging devices.

Currently most PACS's are implemented based on the centralized architecture. Also, in a health region, most PACS's are isolated in the sense that a PACS only manage the internal data. As a result, PACS's are at a small scale and important data is not shared between hospitals. However, as more and different types of advanced modalities are being used, the image data produced by radiology modalities increases rapidly. Larger PACS's will have to be installed to handle this data surge. As the cost to build a large PACS is prohibitively high, it may not be realistic to equip each hospital with a full-fledged PACS. Also, the digital imaging data needs to be shared by clinicians and radiologists across a health region. This is because, first, radiologists and clinicians in different hospitals may need to consult each other. Second, patients may change hospitals. Third, hospitals usually do not have all types of modalities. Fourth, hospitals are not staffed with enough personnel with all specialties. This demands that new PACS's can integrate and manage image data produced within a health region that includes multiple hospitals. For these new PACS's, more often than not the centralized architecture can not provide the required geographical coverage, patient mobility, system scalability, reliability, and especially performance. Therefore, the distributed architecture is the more preferable choice for the PACS administrators.

However, designing and implementing distributed architecture of PACS present many challenges. First, building a PACS archive is very costly and thus multiple archives can easily strain the budget. Hence, the first challenge is to minimize the number of archives in the architecture while still meeting all the predefined targets such as cost and performance. Second, the data processing capacity of an archive is proportional to the cost of the archive. This entails the challenge of determining the capacity of an archive so that the archive is neither over provisioned nor incapable of servicing the users that access this archive. Third, transferring the imaging data over a network can cause long delays as a result of bottlenecks due to either congested bottleneck links or overwhelmed archives, especially when the distributed archives are not wisely placed. The delays will offset the performance gained by investment in archive capacity. The delays may often be unacceptable because of the size and time sensitivity of the imaging data (e.g., in the emergency cases, the response time to retrieve image data of a patient is critical). Thus, where to place the multiple archives in a distributed PACS architecture to best utilize the capacity of the archives and reduce the transferring delay poses another pivotal challenge. Finally, when an image file is produced by a diagnostic device, we also need to choose an archive to accommodate the file so that this file is close to most users and is within the capacity of that archive. The decision of choosing which archive or archives (very large files may be split into parts for storing in multiple archives) to store each single file can be complicated. A file can be stored in a hospital external to the hospital where the file was created and also be accessed by multiple radiologists and clinicians from multiple hospitals.

In summary, to design a distributed PACS architecture, the PACS administrators need to decide the number, the location, and the capacity of the PACS archives and choose which PACS archive(s) to store each study (radiology image and other related data for each diagnostic case) so that they can maximize the performance within a certain budget, or conversely to minimize the cost under a certain performance constraint, e.g., average file transfer time in seconds.

This thesis tackles this problem by making use of a formulation based on mathematical programming. An existing mathematical programming model is first introduced, then modified and solved. The objective is to find the most efficient PACS design that achieves the best performance under a certain budget. The methodology developed can be used to build a software tool that helps PACS system administrators to design a distributed PACS. The methodology can also be helpful to analyze the performance of an existing distributed PACS system that has multiple archives.

The mathematical model includes a performance metric for distributed PACS's. Performance bounds on average file transfer time established by Bonald and Proutière [6] are used to form the objective function. An existing solver for solving optimization problems is used to solve the formulated mathematical programming model.

Like in any model, simplifying assumptions are made when we model the PACS design problem. This may limit the ability to represent an environment as complex as a distributed PACS. Results obtained from solving the mathematical programming model need to be validated. Simulations can be used to validate optimization results [7]. Simulation modelling is able to represent a complex PACS environment that consists of

all archives, network routers, software applications, and network protocols in a more accurate way than mathematical programming modelling. In this research, a discrete-event simulation model is developed to validate the mathematical programming model.

The simulation is at the packet-level. Components such as the network protocols at the end systems, packet-queuing schemes and scheduling in routers, etc., which are left out by abstraction in analytical optimization modelling, are implemented.

When designing the experiments, I selected parameter values for the mathematical programming model and simulation in such a way that the results from the simulation and those from the optimization are comparable. For each set of comparable parameter values, I analyze the two sets of results from optimization and simulation to check if they match.

To illustrate and prove my methodology, I selected two problem instances. One is to design a distributed PACS for the Health Region Authority of a metropolitan city. Traffic data collected from the hospitals in the health region is used. The other problem instance has no counterpart in reality but is created according to the characteristics of the first problem instance. Its scale in terms of number of nodes and network links, however, is twice as large as that of the first problem instance. It is anticipated that these two problem instances encompass a large number of distributed PACS design scenarios.

Simulation results show that the optimal placement produced by solving the optimization model yields good performance. The performance collected from the simulation is close to the objective function value that is obtained from the optimization model solution. The difference between the two is due to a number of realistic components that are missing from the optimization model but are implemented in the simulation model.

To increase the confidence in the proposed approach, experiments were performed with varying budget and network bandwidth, for both the mathematical programming model and the simulation. Results show that (a) with increased budget, more archives may be employed which leads to improved performance, (b) with increased network bandwidth, the performance is also increased. Both are consistent with intuition and are as expected.

Because the mathematical programming model used in this research is nonlinear and integer, the solver only produces local optima. There are multiple algorithms employed by the solver to solve integer nonlinear models. These algorithms have algorithm parameters, which are set to default values at beginning. To search for the parameter settings that produce better results than the default one, I perform experiments with different combinations of parameter values. The experiments show that there is large difference among the objective values produced by different parameter settings. They also show the same pattern for both problem instances with regard to which parameter settings improve the results and which ones degrade them. Moreover, I perform simulation for each solution produced by the solver with different parameter settings. The difference in objective values as a result of different solver parameter settings coincide with the difference in performance obtained from simulation. This further supports the merit of the developed approach. The solver parameter tuning also provides valuable insight into solver parameter settings for solving similar PACS design problems.

In summary, these experiments demonstrate that the mathematical programming model can be used by PACS designers to find a good design for a distributed PACS. In addition, the simulation model developed can be utilized to produce more accurate estimate of the system performance than with the optimization model alone.

This thesis is organized as follows. In Chapter 2, organization of PACS is introduced; previous work that is related to this research is examined.

In Chapter 3 the existing mathematical programming model and its modification are presented. This includes model assumptions, objective function, model parameters, and decision variables. After that, the process to solve the model and to tune the solver parameters is described. Finally, the results of the mathematical programming model are reported and analyzed.

In Chapter 4, results from optimization are validated by simulation. The simulation model as well as experiments and results are presented. Simulation results are analyzed and compared to the results from solving the mathematical programming model.

Finally, Chapter 5 contains the conclusions, a list of thesis contributions, and suggestions for future research.

Chapter 2 Background and Related Work

In this chapter, I first provide a more detailed description of the organization of a PACS, including its components and workflow. Then I introduce some literature related to my research. The literature review focuses on three areas: performance modelling of PACS's, the design of distributed PACS's, and network simulations for performance evaluation.

2.1 Organization of a PACS

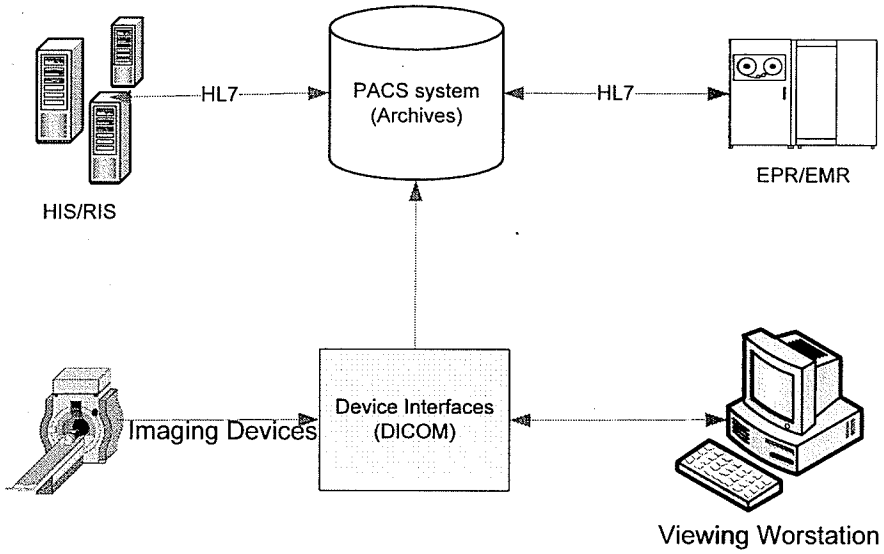
PACS is sometimes referred as image management and communication system (IMAC) also. The main purpose of PACS's is to replace hard-copy (e.g. film) based medical image management with digitalized image managing computer systems. Compared with a legacy film system, PACS has a great multitude of advantages, e.g., easier distribution of images at enterprise level, much shorter image transmission time, simpler archival management with less storage space and personnel, more flexible image manipulation, and easier image copying and backup.

A PACS consists of a number of components interconnected by a broadband network (See Figure 2.1).

In the figure, imaging devices are where the digital diagnostic images are produced. These include all the diagnostic radiology devices, i.e., modalities found in a modern hospital. Modern radiology equipments can produce images in digital form and can be stored in a PACS directly. On the other hand, traditional film producing radiology equipment needs a file digitizer to convert films to digital images. Because of the high

expenses of the modalities, the common practice is that a hospital in a health region is only equipped with certain types of modalities. Modalities are shared among hospitals in a health region.

Figure 2.1 A PACS



PACS system (Archives) store high volume of digital medical data. They include both short-term storage and long-term storage. Short-term storage acts like a local cache, while long-term storage is for priors (historical data). Along with image data the comments and diagnosis from radiologists are also stored in the archives.

Viewing workstations are used by radiologists and clinicians to view the images. It usually has large screen and advanced image processing capabilities. Clinicians mainly retrieve data from PACS while radiologists can make changes to images and write their comments and diagnoses on the images. Clinicians' activities produce read-only traffic.

Radiologists' activities produce writing traffic to data archives in addition to reading traffic.

As an integral part of a healthcare enterprise, PACS interacts with other forms of information systems in the healthcare environment, such as Hospital Information System (HIS), Radiology Information System (RIS) and electronic medical or patient record (EMR or EPR).

Device interfaces (DICOM) shown in Figure 2.1 are the components that integrate imaging devices, modalities, film scanners, from different vendors. The imaging devices are most likely from different manufacturers and not able to communicate with each other. The interfaces make PACS independent of various PACS equipment vendors. Device interfaces implement Digital Imaging and Communications in Medicine (DICOM). DICOM is the de facto standard that defines the data structures and services that are used to store, print and transmit information in medical imaging systems. Two main components of DICOM are image file format definition and network communication protocol [8]. The image file format definition makes possible exchanging files between various devices from different vendors who comply with the standard. The network communication protocol defines an upper level protocol (ULP) over TCP/IP [9]. All systems or equipments that want to be a part of an initiative of Integrating the Healthcare Enterprise (IHE) [10] need to implement the network communication protocol of DICOM [11]. Before DICOM, PACS users were dependent on one vendor because images acquired from a device from one vendor are in proprietary format and can not be understood by devices from another vendor. Nowadays, DICOM has been widely

adopted by hospitals, PACS users, radiology equipment vendors, and gradually by smaller applications like dentists' offices.

HL7 in Figure 2.1 stands for Health Level 7. It provides a standard textual data format to exchange health care information between HIS, RIS and PACS [4]. Usually HIS, RIS and PACS are built upon different computer platforms and they need to convert their internal representation to a standard format or inter-communication. While DICOM facilitates communication between imaging devices and PACS, HL7 is the counterpart between RIS, HIS, EMR and PACS. The difference is that HL7 does not contain communication protocols.

The two standards, DICOM and HL7, have matured since their inception in 1980s. They have made it feasible to integrate heterogeneous health care systems. Both are useful to build a distributed PACS.

In [12], Bandon et al. describe three types of different architectures that could be used to design a PACS. The three types are:

- A centralized system that manages all medical images.
- A hierarchical model with a master PACS connecting multiple mini-PACS's. Each mini-PACS has its specialty, e.g., cardiology, haematology, etc. The master PACS serves as the long-term archive.
- Multiple PACS's. Each PACS is stand-alone, but with different targeted users. There is a search engine for users to query all the data stored in the multiple PACS's.

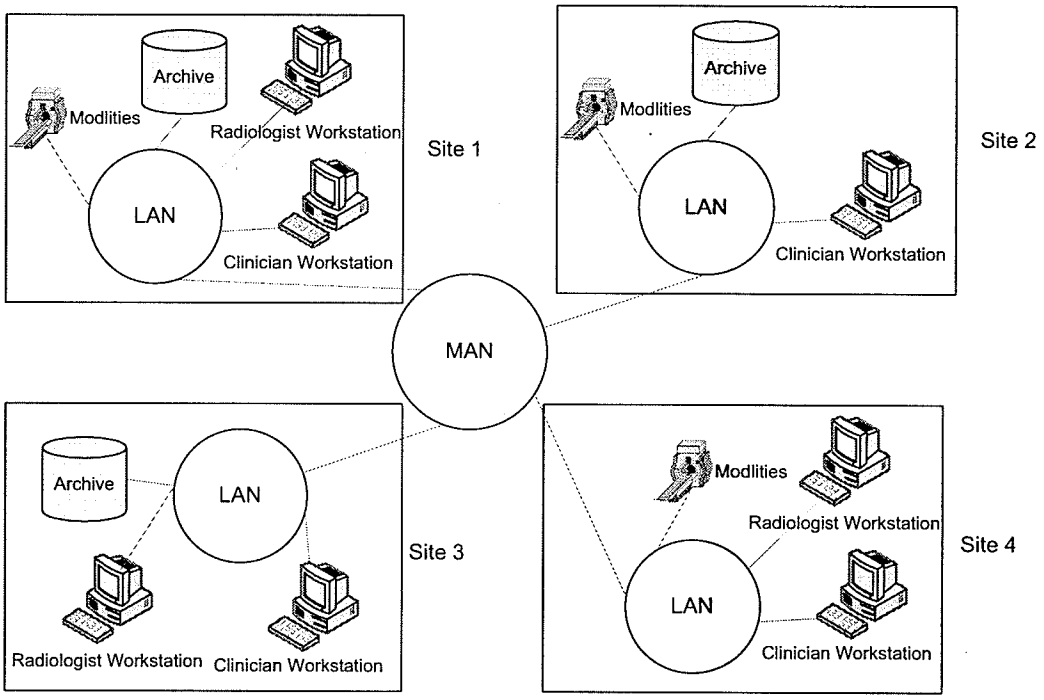
Bandon et al. [12] point out that the third architecture or a hybrid of the second and the third architectures are the future approaches to designing a distributed PACS.

2.2 Distributed PACS

In a distributed PACS within a health region, usually there are multiple archives. These archives often co-locate with hospitals in the region. However, not every hospital is equipped with one due to the high cost of a PACS archive. Archives are equipped for the entire health region and they are interconnected and interdependent. Each of them serves as an integral part to meet the radiology imaging service needs of all the hospitals in the health region. With a distributed PACS, hospitals can share modalities, and expert resources, e.g., radiologists and clinicians. Those hospitals that do not have an on-site PACS archive but have modalities are able to store the image data produced by modalities in a PACS archive located outside the hospital.

Because of the large data volume being transferred in a large area, the components of a distributed PACS are connected by a metropolitan network with broadband links. In some cases, dedicated network links or special network protocols are used to facilitate data transfer and guarantee the response time of data access. The response time or data access performance is the lapse of time from when a data access request is made to when the data access is satisfied. Figure 2.2 shows a distributed PACS with four sites. They are

Figure 2.2 Distributed PACS



connected by a Metropolitan Area Network (MAN). Internal components of each site are connected by a Local Area Network (LAN). The four sites are equipped differently. Site 2 does not have radiologists. Site 3 does not have any modalities. Site 4 does not have archives. Only site 1 is fully equipped.

Sheng and Garcia [1] describe a number of advantages of distributed PACS systems over centralized systems and also presented the issues of designing a distributed PACS system. They identify the issue of database fragment allocation which is the same as deciding archive location and capacity. However, in their research work, they do not propose a solution.

In [13], Huang first describes the "overall view of the current status of enterprise PACS and image distribution". He then reviews three large-scale enterprise PACS's with

multiple archive centres implemented in United States, Korea and Hong Kong. Huang also discusses the advantages and related issues of distributed PACS's. However, Huang does not introduce any optimization methodology when discussing the design of distributed PACS's.

Tsiknakis et al. [14] introduce the architecture of a distributed PACS called TelePACS. The first version of TelePACS has been implemented and installed at the radiology clinic of the University of Athens, various imaging clinics of the University Hospital and the Venizelion Regional Hospital. However, TelePACS distributes the system load by functions. It uses different servers for different functions. For example, it has acquisition servers responsible for the acquisition of medical image data from medical imaging modalities, and archive servers responsible for managing medical image storage. TelePACS does not optimize the location and capacity of the archives so that resources can be best utilized when the number of archives to be built is limited by budget constraints.

The mathematical model of this thesis is based on the one formulated in [15]. The performance metric used by the model in [15] is refined in this thesis by removing the uncertain weight in the objective function [15]. In this thesis, a more complicated problem instance is introduced to show the applicability of the model. Moreover, the parameters of the solver used to solve the model have been tuned to improve the results.

2.3 Modelling of PACS's

The research of Lawrence et al. [16] is one of the early works to introduce the advantages of PACS's over traditional film-based radiological image management. In their research, besides illustrating the concept of a PACS, they also introduce a queuing model approach to analyzing PACS performance. However, their approach deals with a PACS at its early stage when it was in simpler form and at a smaller scale. It does not anticipate the current digital imaging technology and the growing demand to share radiological data.

Chiotis et al. [17] study the performance of a PACS by using a mathematical programming model and the simulation of an M/D/1 queuing system. They present a method to predict the performance of a PACS under various resource requirements. They concluded that simple queuing and simulation models can be efficient methods to obtain PACS performance results. However, their study is on a single server scenario and no PACS distribution and placement is studied.

Huang et al. [18] introduced a testbed for researching the performance of several federated PACS's. In the paper, they describe the challenge of managing large-scale medical image archives. They implement a testbed with each PACS in two clinical sites and one research site. Besides backup procedures and disaster recovery, they discuss the benefits of multiple federated PACS's, e.g., fault-tolerance, performance improvement etc. However, the optimization of the placement of the PACS is not their focus.

In her thesis, Mogatala [7] addresses the web cache location and provisioning issue for Regional Internet Service Providers by building, solving mathematical programming

models and executing discrete event simulations. In the research, Mogatala intends to find the best placement and capacities of web caches that can provide best performance under certain budget. To tackle this problem, Mogatala uses an approach similar to the one I have used. Mogatala first models the issue as a Mixed Integer Programming problem and then solves the mathematical model. After that, Network Simulator 2 (NS2) [19] is used to implement a simulation model to validate the optimization results. The author states that performance improvement can be achieved under certain budget or best investment can be found using the methods above mentioned. However, the problem addressed by Mogatala [7] is linear and the traffic model of the web caching is simpler in that traffic is in one direction directing from web servers to web users. Thus, the optimization and simulation methodologies used by Mogatala are quite different from the ones that I used in this research.

2.4 Network Simulation

Designers of networks want to know the performance of their design of networks before the networks are fully implemented and in production. Experimentation with the physical network is often either infeasible or too costly because of the complexity of the network [20]. This is especially true for building a distributed PACS considering the complexity and cost of building a distributed PACS.

Analytical queuing models have been a common replacement for physical networks to study network performance issues [21]. However, analytical queuing models have drawbacks when the target system to be modelled becomes complex [20]. A detailed list of drawbacks is given by Law and McComas [20]. Some of the drawbacks apply to the

analytical model that is used in this research. They are the potential inaccuracy and uncertainty caused by the assumptions made during modelling abstraction. This makes it necessary to validate the mathematical programming model. Network simulation has been widely used to validate analytical models by researchers who study network performance.

There are two main types of network simulation, packet-level simulation and flow (fluid)-level simulation. The former simulates discrete events of each packet in the network. The latter simulates in a more abstract way in the sense that it only simulates the transfer rate changes of the traffic flows in the network. Packet-level simulation is more accurate but requires many more events be simulated. Thus it has scalability issues when network activities become busier, e.g., when network is large and bandwidth is high [22]. The abstraction from packets to flows can result in large performance gain for flow-level simulation [23]. The performance gain is obtained at the cost of losing information about individual packets and thus accuracy is the cost of flow-level simulation compared to packet-level simulation. Cameron et al. [23] propose an hybrid approach to exploit the accuracy of packet-level simulation and the performance advantage of flow-level simulation by combining packet- and flow- level simulation in a single simulator. However, it brings the challenge of accurately simulating the interaction between flows and packets in the network. Thanks to its accuracy, discrete-event packet-level simulation is used as benchmark to validate other simulation methodologies. Yung et al. [24] use detailed packet-level simulation to validate a simulator they created that integrates a fluid flow model with packet-level simulation. These two types of simulation may be of great help to the designers of distributed systems as they can choose one of the two types of

simulations according to the scale of their system and their preference to accuracy or speed of simulation. For my thesis, I chose packet level simulation for more accuracy after I did preliminary experiments on both packet level and flow level simulation. I implemented flow level simulation using Matlab for one of two problem instances for which I implemented packet level simulation. Results showed that flow-level simulation is indeed faster. But the running time of the packet level simulation is also acceptable for the problem instances I selected. Hence, I chose accuracy over speed in my thesis.

As an example of using simulation to validate analytical models, Monirul et al [25] build a packet-level simulation model to validate their analytical model. The analytical model is to model and analyze virus propagation on Internet. They show that the agreement between the packet-level simulation model and the analytical model give them increased confidence in both models.

Chapter 3 A Mathematical Programming Model for Designing Distributed PACS's

In this chapter, the problem environment for which we need to design a distributed PACS is first described. Next, a key issue, measuring the performance of the distributed PACS, is addressed in regard to the definition of a mathematical programming model of distributed PACS's. Then, an existing analytical model [15] is described and then modified, which includes an objective function, the decision variables and the constraints.

I proceed to solve the analytical model. To illustrate this, I use two health system problem instances. During the process of solving the model, I perform solver parameter tuning experiments to improve the solutions produced by the solver.

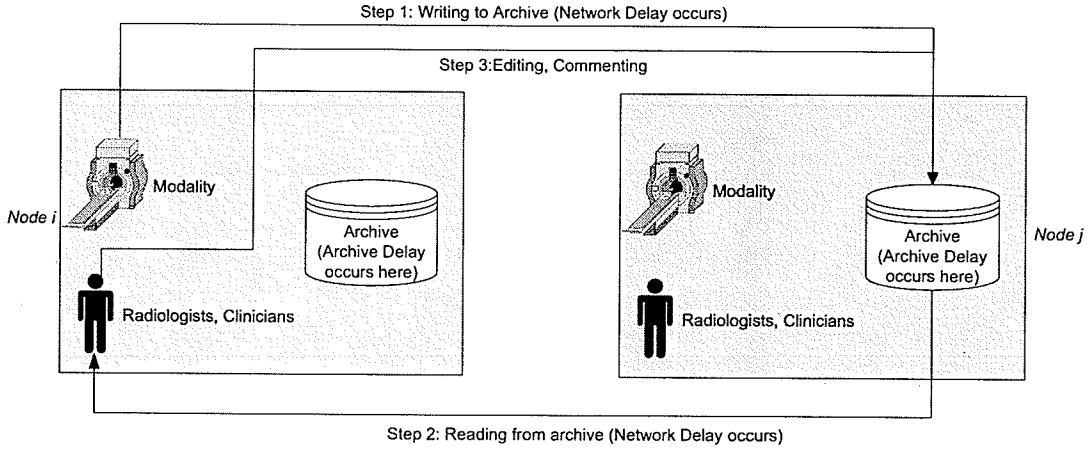
3.1 Problem Environment

The distributed PACS system is developed for a *health region* which includes a set of *sites* such as hospitals, clinics, doctor offices, etc. We consider a health region that has V sites, where each site is a potential location for a PACS archive. Sites are connected by a communication network. In this work, we represent the communication network by the set of sites and a set of L *channels*. A channel l_{ij} is a unidirectional network link that connects sites i to j. Each channel is associated with a capacity B_{ij} which is the bandwidth in Mega bits per second (Mbps) of channel l_{ij}. Routing information of the network is also assumed to be known and fixed which is defined by R_{ijst}:

$$R_{ijst} = \begin{cases} 1, \text{ if route } ij \text{ goes through channel } st \\ 0, \text{ otherwise} \end{cases} \quad i, j \in V, st \in L, \tag{1}$$

where route ij denotes a path from site i to site j that consists of a sequence of channels. Each site may have certain types of *modalities* that produce *radiological studies* which need to be saved in archives (studies are also, in some cases, referred to as *files* or *documents*, or generally as *traffic data)*. Studies are saved in some archives for short-term storage and may be moved to some other archives for long-term storage. The studies in long-term archives are called *priors*. Priors are retrieved when the medical history of patients is requested. Two types of hospital personnel may request the studies and priors. They are clinicians and radiologists.

Figure 3.1 PACS Data Access



In a distributed PACS, the journey of a study usually starts when a modality produces the study and stores it into an archive (step 1 in Figure 3.1). Next, it is read by a radiologist or a clinician (step 2 in Figure 3.1). The radiologist and clinician may edit the study before he/she writes the study back into the archive which causes writing traffic to the

archive (step 3 in Figure 3.1). Consultation between radiologists and clinicians may happen afterwards and cause further reading and writing traffic of the study. During consultation, radiologists and clinicians do not exchange the study directly. The study is always stored into the archive first and then is read by the person whose opinion of the study is requested.

For this problem environment, we assume that we know the following:

- The average rate of studies in Mega bytes per second generated by each modality
- How the clinicians and radiologists access the studies. The information about data access includes the percentage of the studies accessed by the clinicians and radiologists in the same hospital where the studies are produced. It also includes the percentage of the studies accessed across hospitals and consultation among the same group of radiologists that are in different hospitals.

Studies can be classified according to their source and destination [26]. Where a study is produced and where the two types of users, clinicians and radiologists, of a study are located decide the class of study. A study is of *class* (i, j, k) if it is produced at site i, requested by a clinician at site j and a radiologist at site k. i, j, or k can be the same. This notion of class is introduced based on traffic estimations that are collected at hospitals and made available to this study. Based on the two assumptions above, we can calculate, for each class, the rate at which the studies of class (i, j, k) are produced in bytes per second. This rate is denoted as D_{ijk}. We assume that P_{ijkn}, the rate of prior data

of class (i, j, k) that is retrieved to site n, is also known. The detailed explanation of D_{ijk} and P_{ijkn} will be given in Section 3.6.1.

We also assume that the cost of maintaining an archive is linearly proportional to the capacity of the archive. No fixed cost is considered as a part of the total cost of building an archive. Without loss of generality, we assume the cost of one unit of the data processing capacity (1Mbps) is 1. As a result, the total capacity is equal to the budget if a full budget is used.

Table 3.1 summarizes the parameters of a hospital region which we use to define a problem instance in our model. These parameters include the set of sites V, the parameters that define the communication network (which include the set of channels L, the corresponding bandwidth of each channel B, and the routing of the communication network R), the traffic of the network (which include the producing rate of each class of new studies D and the rate of each class of prior studies retrieved to each site P), and the total capacity available to all archives C.

Table 3.1 Parameters of the Mathematical Programming Model

Parameter	Description
V	Set of network sites
L	Set of channels in the network
B_{st}	Bandwidth (in Mbps) of channel st, $st \in L$
R_{ijst}	$=1$ If traffic from site i to j is routed such that it traverses the channel

Parameter	Description
	st , $i,j \in V; st \in L$
	$= 0$ otherwise
D_{ijk}	Rate (in Mbps) of production of class (i,j,k) studies, $i \in V; j,k \in V'$
P_{ijkn}	Retrieval rate of the prior studies of class (i,j,k) to site n (in Mbps) $i,n \in V; j,k \in V'$
C	Total capacity (in Mbps) available for archives, also called budget

Since not every study interests all clinicians or radiologists, a study may not be requested by any clinicians or by any radiologists. We denote by v^0 the null site where no clinicians or radiologists requests a study. For example, studies of class (i,v^0,k) are not requested by any clinicians, studies of class (i,j,v^0) are not requested by any radiologists, and studies of class (i,v^0,v^0) are requested neither by any clinicians nor by any radiologists. In Table 3.1, V' is defined as the union set of V and v^0 , i.e., $V' = V \cup \{v^0\}$.

3.2 Defining the Objective Function

We want a distributed PACS minimizing the delay that users (clinicians, radiologists, etc) experience when accessing studies. Measuring the delay as accurate as possible is critical to the usefulness of the model. Two types of delay that are commonly used in network performance studies are *average delay* and *worst case delay*. The average delay is the

mean value of all delays of the files being transferred. Worst case delay is the longest delay among all delays of the files being transferred [27]. They both have significance to PACS users and are of research values. There are works that focus on studying worst-case delays (for example [28]). Since a constructive way (described later) has been discovered to measure average delay [15], I am going to use average delay, more specifically, the mean time a clinician or radiologist takes to retrieve a unit of data as the performance measure.

The main purpose of this section is to introduce a model of the average delay for distributed PACS's. This model will be used as the objective for the mathematical programming description of distributed PACS's.

3.2.1 Data Access Model

In a distributed PACS, a data access is the transfer of a study from a source site to a destination site. There are two types of data access: data reading and data writing. In a reading access, data is transferred from an archive (source site) to a radiologist or a clinician (destination site). In writing, data is transferred from a modality, a radiologist or a clinician (source site) to an archive (destination site). There is no transfer of data between archives or between users (radiologists or clinicians). There is only one archive involved in each data access. In this thesis, we refer to an ongoing data access as a *flow*. A completed data access is a *data transfer*.

A data transfer consists of two phases. One phase is data being processed at an archive, which can be either a reading or a writing. The other phase is data being transferred in the network. The order of the two phases is dependent on the type of data access. For a data

reading, the archive processing phase is ahead of the transferring phase. For a data writing, it is in the opposite order. Similarly, the total delay of a data access is the sum of two delays. One delay is the time that the archive needs to process the data. This delay is referred to as an *archive delay*. The other delay is the time needed to transfer data on the network. This delay is referred to as a *network delay* or a *transfer delay*. To measure the performance of a distributed PACS, delays on the two phases of a data access are considered.

When calculating the two delays, we consider only transmission delay and queuing delay for both the mathematical model and the simulation model (introduced in Chapter 4). *Transmission delay* defines the amount of time required to load the data to be transferred into the network links [29]. *Queuing delay* is the time when the data is waiting in a buffer before being processed by an archive or transferred by network channels. In this thesis, the archive delay is calculated in the same way as the network delay. This is because

- Both network channels and archives have queuing delays.
- The transmission delay of network channels is similar to the service time of archives.
- Both an archive and a channel can be modelled as a processor-sharing queue.

A process sharing queue defines a system where a server provides an equal fraction of service time to its customers [30]. "Processor-sharing (PS) queues have been widely used in the networking literature to model multiple file transmissions dynamically sharing a fixed amount of bandwidth." [31]

3.2.2 Average Network Delay Model

Measuring the network delay can be very complicated. Network delay of a flow is decided by the bandwidth allocated to the flow. Traffic characteristics or the properties of flows in the network, such as flow size distributions (probabilities of flows being in different possible sizes [32]) and the stochastic process of the new arrival flows (*arrival process* in short), are complex [33]. The scheme to allocate bandwidth to competing flows in a network is a function of traffic characteristics. As a result, the bandwidth allocation scheme is even more complex [34].

Extensive research has been carried out to simplify the performance measurement task, i.e. measuring network delays [6, 34-43]. The main objective has been to identify a method to measure performance that has the insensitivity characteristic, i.e., a performance measurement approach that avoids dealing with most of the complex traffic characteristics.

While trying to address this issue, Bonald and Proutière [37] found that the Whittle queuing network has the insensitivity property. They define the Whittle queuing network as a class of networks whose bandwidth allocation scheme satisfies the following balance property [36]:

> "For all pairs of flows *f, g*, the relative change in the capacity allocated to *f* when *g* is removed is the same as the relative change in the capacity allocated to *g* when *f* is removed."

However, although the insensitivity property of Whittle queuing networks eliminates the complexity of dealing with detailed traffic characteristics, two major obstacles still exist for one to make use of the insensitivity results in order to measure the performance of a PACS network. If one tries to measure the performance of a Whittle queuing network, it would still produce a complex function of the capacity of all channels and the load of all routes [6]. Besides this fact, a PACS network is not necessarily a Whittle queuing network.

A solution to address the first obstacle has been provided by Bonald and Proutière [6]. Assume that a Whittle queuing network is of "balanced fairness" if for any state (number of flows in the network), a route traverses at least one saturated (fully utilized) link [6]. Bonald and Proutière prove that, in a Whittle queuing network of balanced fairness, the mean transfer time (network delay) of a flow is within a range that only depends on per-channel information. This range is given by the following formula:

$$\max_{l \in r} \frac{\eta}{B_l - \mu_l} \le T_r(\eta) \le \sum_{l \in r} \frac{\eta}{B_l - \mu_l} \tag{2}$$

where $T_r(\eta)$ is the mean network delay of a flow of a study of size η on route r on which the flow travels. l is any channel on r, and B_l, μ_l denote the bandwidth and demand of l respectively. The term on the left side is the lower bound of $T_r(\eta)$ which is the η divided by the smallest bandwidth slack of all the channels on the route. The term on the right side is the upper bound of $T_r(\eta)$ which is the summation of η divided by the bandwidth slack of each channel on the route. Equation (2) clearly shows that the performance (the mean network delay) depends only on the demand and the capacity of

the network channels and is insensitive to specific traffic details such as flow size distributions, the arrival process of flows, etc.

In [39], Timonen shows that the above mentioned results of performance bounds are not restricted only to networks of balanced fairness. He showed, through experimentations, that the fairness scheme of a network has little impact on the performance. The fairness schemes that he studied and simulated include common fairness schemes such as max-min [44], proportional [41] and α-fairness [45]. The prevalent TCP protocol used by PACS networks most likely uses one of these fairness schemes to allocate bandwidth between flows [29]. Based on this discussion, it is therefore safe to use the two bounds given by Bonald and Proutière [6] for measuring the performance of a PACS network.

We could use the lower bound or the upper bound given in Equation (2) as the objective function. Since the objective is to minimize the mean file access time of distributed PACS's, it is more important to know the upper bound. Hence, we use the upper bound for the objective function. Moreover, after knowing one bound, we can calculate the other bound according to the formula of the two bounds shown in Equation (2) using the bandwidth and load of each channel. The load of each channel is part of the solution of the mathematical programming model.

Adding the archive and the network delay, the upper bound of the mean delay of a flow of size η on route r is:

$$\sum_{l \in r} \frac{\eta}{B_l - \mu_l} + \frac{\eta}{z_m - v_m} \tag{3}$$

where m is the archive on route r, z_m is the capacity of archive m connected to route r in Mbps and v_m is the load on archive m in Mbps. In the formula, the first term is the mean network delay, the second the archive delay.

Considering we want to minimize the mean data access delay of all routes in a distributed PACS, this objective can be expressed by summing up the delay of a flow of size η on every route in the network:

$$\sum_{r \in N} \left(\sum_{m \in r, l \in r} \frac{\eta}{B_l - \mu_l} \right) + \sum_{m \in V} \frac{\eta}{z_m - v_m} \tag{4}$$

in which N is all the routes in the network.

The objective function is based on this formula (refer to Section 3.4 for the objective function). The objective function is nonlinear because in the denomination μ is a variable.

3.3 Other Decision Variables and Constraints

From Equation (4), we can see the objective function value is decided by the bandwidth slack of each channel, i.e., the difference between the bandwidth B_l and the load μ_l of the channel, and the capacity slack of each archive, i.e., the difference between the capacity z_m and the load v_m of the archive. The bandwidth slack of any channel st is decided by the load on st, μ_{st}, because the bandwidth of each channel is known. The load on a channel is the summation of the rate of the data that goes through the channel, including the reading and the writing data rate. Equation (5) shows how μ_{st} is calculated.

$$\mu_{st} = \sum_{i,j \in V} R_{ijst}\left(r_{ij} + w_{ij}\right) \qquad \forall st \in L \tag{5}$$

where r_{ij} is the rate of the data reading from site i (where the archive is located) to j in Mbps, w_{ij} is the rate of the data writing to site j (where the archive is located) from i in Mbps and R_{ijst} is as defined in Table 3.1.

r_{ij} is the total data rate of all the new studies that are stored at the archive at site i and are retrieved to site j plus the total data rate of all the priors that are stored at the archive at site i and are retrieved to site j. According to the definition of the study class, study classes (k,l,j) and (k,j,l) are retrieved by either radiologists or clinicians at site j, $k \in V, l \in V'$. Assume variable x_{ijkm} decides the percentage of the studies of class (i,j,k) that are stored at the archive at site m where $m \in V$. Then $\sum_{k \in V; l \in V'} \left(D_{klj} x_{klji} + D_{kjl} x_{kjli}\right)$ is the total data rate of all the new studies that are stored at the archive at site i and are retrieved to site j. $\sum_{k \in V; l,m \in V'} P_{klmj} x_{klmi}$ is total the data rate of all the priors that are stored at the archive at site i and are retrieved to site j. Equation (6) shows that how r_{ij} is calculated.

$$r_{ij} = \sum_{k \in V; l,m \in V'} P_{klmj} x_{klmi} + \sum_{k \in V; l \in V'} \left(D_{klj} x_{klji} + D_{kjl} x_{kjli}\right) \qquad \forall i,j \in V \tag{6}$$

Similarly, w_{ij} is the total data rate of the new studies that are stored to the archive at site j from site i. w_{ij} does not involve priors. Equation (7) shows that how w_{ij} is calculated.

$$w_{ij} = \sum_{k,l \in V'} \left(D_{ikl} x_{iklj} \right) \qquad \forall i, j \in V \tag{7}$$

The load of a channel can not exceed the bandwidth of the channel. Equation (8) defines this constraint.

$$\mu_{st} < B_{st} \qquad \forall st \in L \tag{8}$$

As for the archive capacity slack in Equation (4), the load of an archive, v_m, is the total data rate reading from the archive and the total data rate writing to the archive. Equation (9) shows how v_m is calculated.

$$v_m = \sum_{j \in V} (w_{jm} + r_{mj}) \qquad \forall m \in V \tag{9}$$

The load of an archive can not exceed the capacity of the archive. Equation (10) defines this constraint.

$$v_m < z_m \qquad \forall m \in V \tag{10}$$

The summation of the percentage of a class of studies stored in each archive is equal to 100%. Equation (11) defines this constraint.

$$\sum_{l \in V} x_{ijkl} = 1 \qquad \forall i \in V, j, k \in V' \tag{11}$$

Also, the summation of the capacity of all archives can not exceed the total capacity (budget). Equation (12) defines this constraint.

$$C \geq \sum_{m \in V} z_m \tag{12}$$

Table 3.2 lists the decision variables.

Table 3.2 Decision Variables of the Mathematical Programming Model

Variable	Description
z_m	Capacity of archive $m \in V$
x_{ijkm}	Percentage of studies of class (i,j,k) that are stored in the archive at site m, $0 \le x \le 1$, $i,m \in V; j,k \in V'$

Table 3.3 lists the supplemental decision variables of the model. The supplemental decision variables are dependent on or constrained by the decision variables. In other words, once the values of the decision variables in Table 3.2 are determined, the values of the supplemental decision variables are determined.

Table 3.3 Supplemental Decision Variables

Variable	Description
r_{ij}	The rate of studies reading from archive i to j in *Mbps*, $i,j \in V$
w_{ij}	The rate of studies writing to archive j from i in *Mbps*, $i,j \in V$
μ_{st}	Load on the channel st in Mbps, $st \in L$
v_m	Total load on the archive at site m in Mbps, $m \in V$

3.4 The Mathematical Programming Model

From Equation (4), we can see the objective function value is also decided by the size of all studies transferred. Assume we want to minimize the average transfer time of one unit data which is defined as the amount of data transferred in one second in the target distributed PACS. The size of one unit data can be calculated using:

$$\eta_{ij} = \begin{cases} r_{ij} \times 1 & \text{if i is the archive, i,j} \in \text{V} \\ w_{ij} \times 1 & \text{if j is the archive, i,j} \in \text{V} \end{cases} \tag{13}$$

The size of one unit of data is different for different transfer routes. If a route starts with an archive, $\eta_{ij} = r_{ij}$. If a route ends with an archive, $\eta_{ij} = w_{ij}$.

With the previously described information, the mathematical programming model can be described as follows:

Plugging in the variables and parameters into Equation (4), the first term $\sum_{r \in N}\left(\sum_{m \in r, l \in r} \frac{\eta}{B_l - \mu_l}\right)$ is equal to $\sum_{i,j \in V}\left(\left(r_{ij} + w_{ij}\right)\sum_{st \in L} \frac{R_{ijst}}{B_{st} - u_{st}}\right)$. The second term $\sum_{m \in V} \frac{\eta}{z_m - v_m}$ is equal to $\sum_{i,j \in V}\left(\frac{r_{ij}}{z_i - v_i} + \frac{w_{ij}}{z_j - v_j}\right)$. Adding these two terms together, we have this objective function shown in Equation (14).

$$Objective = \min\left\{\sum_{i,j \in V}\left(\frac{r_{ij}}{z_i - v_i} + \frac{w_{ij}}{z_j - v_j} + \left(r_{ij} + w_{ij}\right)\sum_{st \in L} \frac{R_{ijst}}{B_{st} - u_{st}}\right)\right\} \tag{14}$$

Subject to:

$$r_{ij} = \sum_{k \in V; l, m \in V'} P_{klmj} x_{klmi} + \sum_{k \in V; l \in V'} \left(D_{klj} x_{klji} + D_{kjl} x_{kjli} \right) \qquad \forall i, j \in V \tag{15}$$

$$w_{ij} = \sum_{k,l \in V'} \left(D_{ikl} x_{iklj} \right) \qquad \forall i, j \in V \tag{16}$$

$$u_{st} = \sum_{i,j \in V} R_{ijst} \left(r_{ij} + w_{ij} \right) \qquad \forall st \in L \tag{17}$$

$$u_{st} < B_{st} \qquad \forall st \in L \tag{18}$$

$$v_m = \sum_{j \in V} (w_{jm} + r_{mj}) \qquad \forall m \in V \tag{19}$$

$$v_m < z_m \qquad \forall m \in V \tag{20}$$

$$C \geq \sum_{m \in V} z_m \tag{21}$$

$$\sum_{l \in V} x_{ijkl} = 1 \qquad \forall i \in V, j, k \in V' \tag{22}$$

$$x_{ijkl}, r_{ij}, w_{ij}, u_{ij}, v_j, z_j \geq 0 \qquad \forall i, j, k, l \in V \tag{23}$$

Equations (15) and (16) define reading and writing traffic respectively by congregating classes of studies that go through route ij. Equation (17) defines the load on channel st which is the summation of all flows on the routes that pass channel st. Equation (18) specifies that load on each channel should be less than the channel bandwidth. Equation (19) defines the load on an archive as the summation of all reading traffic leaving that archive and writing traffic arriving at that archive. Equation (20) specifies load on an archive should not exceed its capacity. Equation (21) makes sure the summation of the capacity of all archives is under the total capacity.

3.5 Solving the Model

To solve the above mathematical programming problem, I use an existing solver, a software package called KNITRO [46]. KNITRO is commercially available but also free under some restrictions. I use the free version of KNITRO provided by the optimization server NEOS (Network Enabled Optimization System) [47]. NEOS is maintained by the Optimization Technology Center, a joint venture of Argonne National Laboratory and Northwestern University in USA. The solvers supported by NEOS are kept state-of-the-art [48].

To submit an optimization problem to a solver, we need to describe the mathematical model in a language that the solver understands. There are several languages available such as GAMS, MGG, LINGO, MPL, AIMMS, and AMPL [49]. AMPL is the most widely accepted language among the optimization solvers. It has a natural syntax and general set and indexing expressions [50] which provide efficiency to implement a mathematical programming model. Also, AMPL is supported by KNITRO and NEOS. Therefore, I use AMPL to submit my optimization problem to the solver KNITRO.

3.5.1 Solver Characteristics

The KNITRO solver is mainly designed for solving large-scale general nonlinear optimization problems [51]. Nonlinear optimization problems are hard to be solved exactly in a reasonable time; solutions provided by algorithms for these problems only approximate the optimal solution. Furthermore, performance of algorithms for solving these problems varies substantially from one problem to another, even from one problem

instance to another. To obtain more confidence in the solution process, one often tries different algorithms for a same problem. Finally, each algorithm usually has several input parameters which adapt the algorithm to some known specific characteristics of a particular nonlinear optimization problem.

The KNITRO solver systematizes in a single software the above ad hoc solving process. KNITRO contains three algorithms for solving nonlinear optimization problems: one Active Set algorithm and two interior-point algorithms, Interior/Direct and Interior/CG [51]. KNITRO also includes several parameters for adapting the algorithms to different problems and problem instances. In this section, we only describe a subset of KNITRO parameters. Those are the parameters we have used for solving the model in Section 3.4. These parameters are listed below together with a brief description of their functionality and default values.

The first and most important input parameter (*alg*) is the one that specifies which algorithm is used for solving the optimization problem. The default value of this parameter, 0, will cause the solver to choose the algorithm for the user based on the characteristics of the input problem.

Parameter "*bar_murule*" controls the strategy that the solver uses to modify the barrier value for barrier algorithms (the two Interior Point algorithms). This parameter does not apply to Active Set algorithm. The default value of this parameter, 0, will cause the solver to choose the barrier modifying strategy automatically.

Parameter "*hessopt*" specifies how the "*Hessian of the Langrangian*" [51] (page 57) is computed. When "*hessopt*" is set to 6, it works with another parameter "*lmsize*" to let the

solver compute the size of the memory used by the quasi-Newton BFGS Hessian [51] which are used by all three algorithms.

Parameter "*maxit*" specifies that the maximal number of iterations (an iteration is an attempt to search the next solution from the current point according to a certain algorithm) that the solver attempts to find the best solution before the solver exits [51]. It defaults to 10,000.

Parameter "*multistart*" controls if the solver uses multiple start points. Its default value is 0, which means that a single start point will be used. When it is set to 1, which means that the solver will use multiple start points, another parameter "*ms_maxsolves*" controls how many start points the solver tries. When "*ms_maxsolves*" is set to 0, the solver will choose the number of start points to be *min(200, 10N)* where N is the number of variables in the problem.

The parameters of my focus and their meaning are listed in Table 3.4 [51].

Table 3.4 Description of Important KNITRO Solver Parameters

Parameter	**Description**	**Default Value**
alg	Indicates which algorithm to use to solve the problem 0 (auto): Let KNITRO automatically choose an algorithm, based on the problem characteristics. 1 (direct): Use the Interior/Direct algorithm. 2 (cg): Use the Interior/CG algorithm. 3: Active algorithm	Default: 0

Parameter	Description	Default Value
bar_murule	barrier parameter update rule: 0: automatic barrier rule chosen 1: monotone decrease rule 2: adaptive rule 3: probing rule 4: safeguarded Mehrotra predictor-corrector type rule 5: Mehrotra predictor-corrector type rule 6: rule based on minimizing a quality function	Default: 0
hessopt	Specifies how to compute the (approximate) Hessian of the Lagrangian. 1: compute exact Hessian (User provides a routine for computing the exact Hessian). 2: compute (dense) quasi-Newton BFGS Hessian 3: compute (dense) quasi-Newton SR1 Hessian 4: compute Hessian-vector products using finite differencing 5: exact Hessian-vector products 6: use limited-memory BFGS Hessian approximation 6 (lbfgs): KNITRO computes a limited-memory quasi-Newton BFGS Hessian (its size is determined by the option "lmsize").	Default: 1
lmsize	Specifies the number of limited memory pairs stored when approximating the Hessian using the limited-memory quasi-Newton BFGS option. The value must be between 1 and 100 and is only used when "hessopt=6". Larger values may give a more accurate, but more expensive, Hessian approximation. Smaller values may give a less accurate, but faster, Hessian approximation.	Default: 10

Parameter	Description	Default Value
maxit	Specifies the maximum number of major iterations before termination.	Default: 10000
ms_maxsolves	Specifies how many start points to try in multi-start. 0: Let KNITRO automatically choose a value based on the problem size. The value is min(200, 10N), where N is the number of variables in the problem.	Default: 0,
multistart	Indicates whether KNITRO will solve from multiple start points to find a better local minimum. 0 (no): KNITRO solves from a single initial point. 1 (yes): KNITRO solves using multiple start points. ms_maxsolves option has no effect unless "multistart =1".	Default: 0

3.6 Solver Parameters Tuning

The parameters tuning aims at finding which parameter setting combinations produce the best solutions. Since the problem is modelled as nonlinear, the solver KNITRO only returns a local optimum [48], different parameter settings may return different local optima. For example, choosing different initial points will cause the solver to explore different regions of the solution space and to return different local optima. Moreover, as explained above, it is difficult to determine which algorithm in KNITRO will yield the best results for a particular problem. In addition, for most parameters it is difficult to determine which value produces best results [51]. Hence, parameter tuning for the solver is necessary to obtain better solutions than those obtained with the default parameter settings. The parameter settings (of the solver) that yield better solutions in a reasonable

time will be used when the mathematical programming model is being solved for different scenarios such as different problem instances or different input parameters.

One way to obtain the best possible solutions for any problem instance will be to run the solver for the problem instance with all the possible combinations of the parameter values. This is of course very inefficient, if possible at all.

Rather, one seeks to find the best possible parameter setting for a small subset of the problem instances which best represent the whole domain of the problem. Borrowing the methodology used by Steven et al. [52], I first select two problem instances that best represent the problem of archive planning for distributed PACS's. Second, I perform some preliminary experiments to single out the parameter settings that impact the optimization results the most. Third, I perform factorial experiments for the chosen parameter settings.

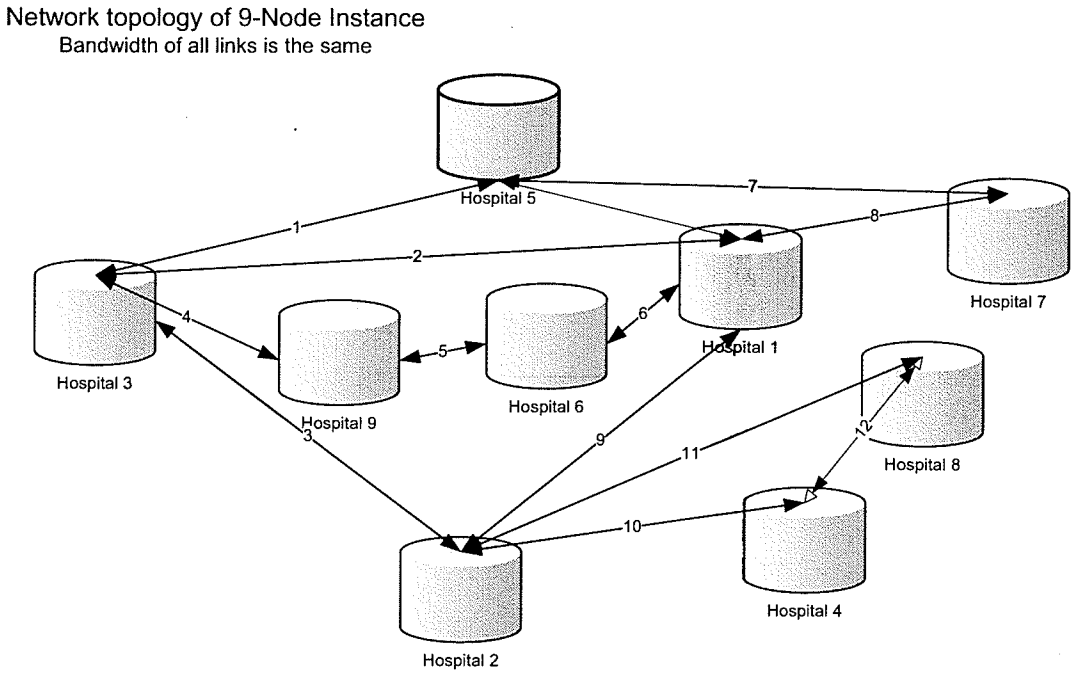
3.6.1 Selected Problem Instances

When choosing the problem instances and constructing the instance environment, we try to choose the problem instances that well represent the distributed PACS designing challenges in terms of scale (number of sites) and data access pattern. Multiple problem instances increase our confidence in our methodology, but this also increases the number of experiments to carry out and the amount of data to analyze. For tuning the parameters, I have chosen to use two problem instances, one has 9 sites, and the other has 15 sites. The two problem instances cover the scale of major real problem instances of distributed PACS design [4].

3.6.1.1 9-Site Problem Instance

This problem instance is based on a health region in a metropolitan city in Canada. The network topology shown in Figure 3.2 is in the network design proposed for the health region in [53]. A node in the network corresponds to a hospital site.

Figure 3.2 9-Site Instance Network Topology



There are 9 sites and 12 links (24 channels) in this network.

Scale between hospitals are quite different [54]. As you can see from the Figure 3.2, sites 1, 2 and 3 are the hubs, i.e. they have more hospitals connected to them. They are the largest in scale in terms of equipments, patient beds and staff etc. Studies produced by modalities located at smaller hospitals will likely be reviewed by the specialists in larger hospitals. On the other hand, modalities in larger hospitals will most likely be referred by

the clinicians or physicians in smaller hospitals. These two types of activities cause cross-hospital data transfer. However, compared to the amount of data accessed locally, cross-hospital data transfer due to modality share or cross consultation between radiologists is less [54].

I gathered data for the parameters in Table 3.1 from a few sources. Information about how the clinicians and radiologists access data has been obtained from Camorlinga and Schofield [26]. The information includes the percentage of the studies accessed by the clinicians and radiologists in the same hospital and the percentage of the studies accessed across hospitals. This source also provided the information about the ratio of the amount of prior studies accessed daily to the amount of new studies generated daily. An estimation about the amount of studies that is produced at each site daily has been obtained from Otukile [53]. In summary, the following information is given:

- M_i: The amount of studies in Megabytes produced per day at site i
- P_{ij}^R: The fraction of studies that are produced in site i and are requested by radiologists at site j
- P_{ij}^C: The fraction of studies that are produced in site i and are requested by clinicians at site j
- R_{in}: The fraction of new studies that are produced at site i and are sent as priors to site n.

Using the above information, D_{ijk} (See Table 3.1 for the meaning) in the unit of Mbps can be calculated using Equation (24):

$$D_{ijk} = \frac{M_i \times 8}{3600 \times 8} P_{ij}^R P_{ik}^C \tag{24}$$

3600×8 is the number of seconds in the working hours of a day.

P_{ijkn} (See Table 3.1 for the meaning) in the unit of Mbps can be calculated using Equation (25):

$$P_{ijkn} = D_{ijk} \times R_{in} \tag{25}$$

We categorize traffic of a distributed PACS as source destination pairs. Each different class of studies has either a different source or a different destination site. Since there are nine sites in the network, there are $9^3 = 721$ possible different classes of studies. Since there are no routes between some sites or no studies transferred between some of the hospitals, the actual number of study classes is less than 721. That means there are D_{ijk} values that are 0. The arrival process of each class of studies is assumed to be Poisson.

3.6.1.2 15-Site Instance

To explore the impact of the solver parameters on the optimization results and applicability of the optimization and simulation model to other problem instances, I create another problem instance.

Figure 3.3 15-Site Instance Network Topology

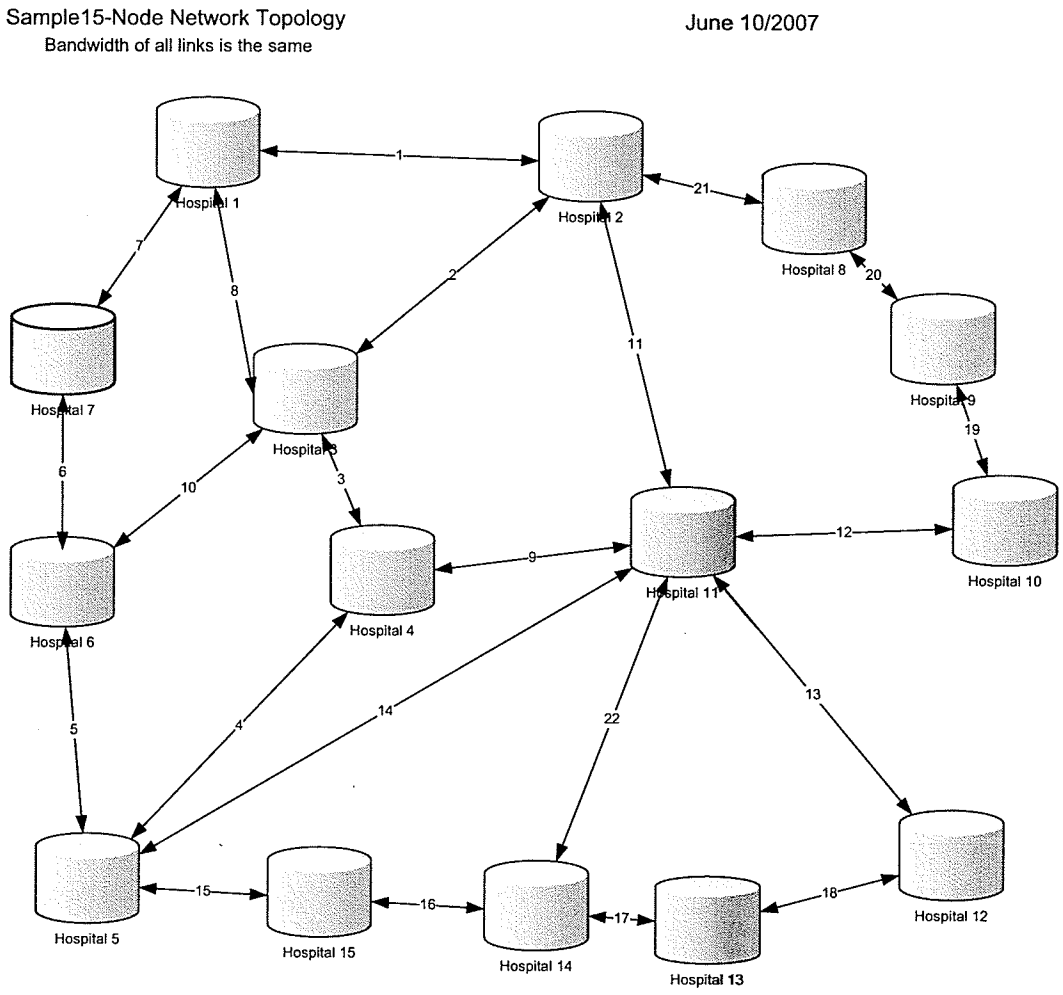


Figure 3.3 shows the network topology. In this network, there are 15 sites, 22 links, and 44 channels.

Unlike the 9-Site instance, I use a different approach to generate D_{ijk} for this 15-Site instance. I generate D_{ijk} using an exponential distribution function with a mean rate. The mean rate is chosen to make sure there is enough traffic to show the effectiveness of the optimization and not too much traffic that will overwhelm the system. The percentage of

D_{ijk} values that are zero is set to the same as that of 9-site instance. When generating D_{ijk}, I set some D_{ijk} to zero according to a uniformly distributed random variable.

To calculate P_{ijkn}, I use the same R_{in} and the same formula, Equation (25), as I used for 9-site instance.

The remaining parameters for this problem instance are set the same way as the previous instance only different in scale.

For all the experiments performed for solver parameter tuning, the parameters for the mathematical model are fixed. For 9-site instance, C=50Mbps and B=1Mbps. For 15-site instance, C=1500Mbps and B=100Mbps.

3.6.2 Preliminary Experiments

Preliminary experiments are to weed out parameter settings that do not improve the optimization results so that the number of experiments in later factorial design is reduced. For each parameter of the solver, if it accepts a continuous range, I experiment with 2 end values. If it accepts discrete values, I experiment all of them. The largest number of discrete parameter values is 7. If it accepts open end values, I experiment the smallest and largest value that does not cause the experiment timeout and produces a feasible solution.

After having performed the above preliminary experiments, I have the following results and conclusions:

- The results in Table 3.5 show that when applied to the 9-site instance and 15-site instance, algorithm 3 (Active Set) produces much worse objective value and also

takes much longer time (10 times as much) than does the Interior/Direct algorithm. This corroborates the statement in the manual that Active Set algorithm is not so efficient for large problem instances as the other 2 algorithms unless there is some prescient knowledge about a good start solution [51] (page 52). Therefore, I select the first 2 algorithms for the factorial experiment design.

Table 3.5 Preliminary Experiments for Different Algorithms

9-Site Instance			**15-Site Instance**		
Algorithm	Objective Value	Solve Time (seconds)	Algorithm	Objective Value	Solve Time (seconds)
alg=1	1.97	4.12	alg=1	0.1782	24.96
alg=2	1.53	12.92	alg=2	0.0977	449.21
alg=3	11561.67	156.35	alg=3	352145.15	5012.98

Note: The solve time is the time from when the experiment is started to when the experiment is terminated.

- Avoid using hessopt=2 or 3, they use up memory and produce "Not enough Memory error", the solver manual advices use these 2 values for small problems (the number of variables is less than 1000). Hence, these two values are removed from the later factorial experiment design.

- When *lmsize* is set to default value or larger than 10, most of the time the solver times out before the solution converges. For the factorial experiment design, I choose 5 for *lmsize*.

- In Table 3.6, I present results of testing the impact of parameter *ms_maxsolves*. The experiments are performed on both instances for 3 different *bar_murule* values. Varying *bar_murule* value is to show the impact of parameter *ms_maxsolves* under different circumstances. From the results, we can see that multiple start points improve the optimization results although a small increase from 0 to 10 does not. Another fact one can observe from the results in the table is that increasing the start points also multiples the solve time. Note: when $ms\_maxsolves = 0$, the actual number of initial points is min(200, $10N$). Since N is in order of 10^3 and 10^4 respectively for the 9-site and the 15-site instance, the solver will always choose 200. For the factorial experiment design, I choose 0, 1, and 10 for the *ms_maxsolves* parameter.

Table 3.6 Impact of ms_maxsolves

Other parameters: alg=1, hessopt=1, multistart=1

bar_murule	ms_maxsolves	9-Site Instance		15-Site Instance	
		Objective Value	Solve Time (Seconds)	Objective Value	Solve Time (Seconds)
1	0	1.183209	730.550	0.061	7202.760
	1	1.979603	6.644	0.165	25.470
	10	1.979603	33.882	0.165	247.291
3	0	1.170	760.984	0.056	6554.022

bar_murule	ms_maxsolves	9-Site Instance		15-Site Instance	
		Objective Value	**Solve Time (Seconds)**	**Objective Value**	**Solve Time (Seconds)**
	1	1.255	5.376	0.112	45.427
	10	1.255	35.194	0.112	320.028
4	0	1.171	1865.200	0.056	10011.520
	1	1.252	16.810	0.119	47.199
	10	1.252	64.770	0.119	544.730

- In most experiments the solution searching terminates within 50 iterations. The largest number of iterations in all the experiments that I have done for both problem instances is 450. The default value of parameter *maxit* is 10000. The experiment results show that increasing the *maxit* up from 10000 indeed does not change the result. Hence, I use the default value and remove *maxit* from the later factorial experiment design.

3.6.3 Factorial Experiment

Factorial experiments are those experiments to test the factorial combinations of the solver parameters settings selected before. According to results from preliminary experiments, I choose the following parameters and their values for factorial experiments:

$$
\begin{aligned}
&alg = \{1,2\} \\
&barrule = \{1,2,3,4,5,6\} \\
&hessopt = \{1,4,5,6\} \ and \ lmsize = 5 \\
&multistart = 0; multistart = 1 \ and \ ms\_maxsolves = \{0,1,10\}
\end{aligned}
\tag{26}
$$

The number of full factorial combinations with the above parameter settings is 80, excluding some parameter values that conflict with each other. *barrule 3, 4, 5, 6* are available only to the Interior/Direct Algorithm [51] (page 35). *hessopt 4, 5* are not available to the Interior/Direct Algorithm [51] (page 38).

3.6.4 Solver Tuning Results and Conclusions

Table 3.7 to Table 3.10 show the results of solver parameter tuning. Results in Table 3.7 and Table 3.8 are from the experiments performed for 9-site instance, Table 3.9 and Table 3.10 for 15-site instance. In these tables, the first column is the parameter setting index. The same parameter setting of each experiment has the same index number for both instances. The same index number provides convenience for comparing the impact of a parameter setting on both problem instances. Because two problem instances have different order for the results of the same parameter settings, the index column are not in order.

The objective value from each experiment is converted to the average time to transfer one Mega bits (Mb) data so that later it is easier to compare the results with the simulation results. To get the average time to transfer one Mb of data, the objective value is divided by the total size of the data transferred. The total size is the total data rate times 1 second (refer to the definition of the objective function in Section 3.2), which is equal to the

summation of the reading and writing data rates. The lower bound column in the tables is calculated using the formula of lower bound shown in Equation (2) and the solution of solving the model.

To show which parameter settings result in the best performance and which parameter settings result in the worst performance, the top and bottom results for the 2 problem instances are shown. Results in the tables are ordered by the objective values listed in the upper bound column in the tables as the upper bound values are our main concern. Table 3.7 shows the top results among all the experiments performed for the 9-site instance, Table 3.8 the bottom results. Table 3.9 shows the top results among all the experiments performed for the 15-site instance, Table 3.10 the bottom results.

Table 3.7 Top Results for the 9-Site Instance

No.	Solver Parameters	Lower Bound	Upper Bound	Solve Time(s)
42	alg=1, bar_murule=6, hessopt=1, multistart=1, ms_maxsolves=0	1.0226	1.1691	1045.17
10	alg=1, bar_murule=2, hessopt=1, multistart=1, ms_maxsolves=0	1.0228	1.1699	774.32
18	alg=1, bar_murule=3, hessopt=1, multistart=1, ms_maxsolves=0	1.0228	1.1699	760.98
34	alg=1, bar_murule=5, hessopt=1, multistart=1, ms_maxsolves=0	1.0236	1.1709	1008.33
26	alg=1, bar_murule=4, hessopt=1, multistart=1, ms_maxsolves=0	1.0236	1.1710	865.20

No.	Solver Parameters	Lower Bound	Upper Bound	Solve Time(s)
70	alg=2, bar_murule=2, hessopt=4, multistart=1, ms_maxsolves=0	1.0237	1.1711	9777.22
46	alg=1, bar_murule=6, hessopt=6, lmsize=5, multistart=1, ms_maxsolves=0	1.0262	1.1754	10230.73
2	alg=1, bar_murule=1, hessopt=1, multistart=1, ms_maxsolves=0	1.0331	1.1832	730.55
54	alg=2, bar_murule=1, hessopt=4, multistart=1, ms_maxsolves=0	1.0353	1.1860	2553.04
58	alg=2, bar_murule=1, hessopt=5, multistart=1, ms_maxsolves=0	1.0353	1.1860	3437.35
50	alg=2, bar_murule=1, hessopt=1, multistart=1, ms_maxsolves=0	1.0488	1.2037	2907.81
25	alg=1, bar_murule=4, hessopt=1, multistart=0	1.0864	1.2519	14.03

Table 3.8 Bottom Results for the 9-Site Instance

No.	Solver Parameters	Lower Bound	Upper Bound	Solve Time(s)
61	alg=2, bar_murule=1, hessopt=6, lmsize=5, multistart=0	1.6981	2.0069	3261.16
63	alg=2, bar_murule=1, hessopt=6, lmsize=5, multistart=1, ms_maxsolves=1	1.6981	2.0069	3072.84

No.	Solver Parameters	Lower Bound	Upper Bound	Solve Time(s)
64	alg=2, bar_murule=1, hessopt=6, lmsize=5, multistart=1, ms_maxsolves=10	1.6981	2.0069	5739.67
45	alg=1, bar_murule=6, hessopt=6, lmsize=5, multistart=0	1.8090	2.1474	299.30
47	alg=1, bar_murule=6, hessopt=6, lmsize=5, multistart=1, ms_maxsolves=1	1.8090	2.1474	105.09
48	alg=1, bar_murule=6, hessopt=6, lmsize=5, multistart=1, ms_maxsolves=10	1.8090	2.1474	693.50
62	alg=2, bar_murule=1, hessopt=6, lmsize=5, multistart=1, ms_maxsolves=0	8.0731	9.3137	5752.42
77	alg=2, bar_murule=2, hessopt=6, lmsize=5, multistart=0	415.5451	435.8329	5076.42
78	alg=2, bar_murule=2, hessopt=6, lmsize=5, multistart=1, ms_maxsolves=0	415.5451	435.8329	10188.88
79	alg=2, bar_murule=2, hessopt=6, lmsize=5, multistart=1, ms_maxsolves=1	415.5451	435.8329	5110.12
80	alg=2, bar_murule=2, hessopt=6, lmsize=5, multistart=1, ms_maxsolves=10	415.5451	435.8329	10102.67

Table 3.9 Top Results for the 15-Site Instance

No.	Solver Parameters	Lower Bound	Upper Bound	Total Solve Time(s)

No.	Solver Parameters	Lower Bound	Upper Bound	Total Solve Time(s)
10	alg=1, bar_murule=2, hessopt=1, multistart=1, ms_maxsolves=0	0.0293	0.0560	3668.06
18	alg=1, bar_murule=3, hessopt=1, multistart=1, ms_maxsolves=0	0.0293	0.0560	6554.02
42	alg=1, bar_murule=6, hessopt=1, multistart=1, ms_maxsolves=0	0.0307	0.0562	9976.39
34	alg=1, bar_murule=5, hessopt=1, multistart=1, ms_maxsolves=0	0.0308	0.0564	8020.44
26	alg=1, bar_murule=4, hessopt=1, multistart=1, ms_maxsolves=0	0.0308	0.0564	10011.52
54	alg=2, bar_murule=1, hessopt=4, multistart=1, ms_maxsolves=0	0.0341	0.0610	10260.70
58	alg=2, bar_murule=1, hessopt=5, multistart=1, ms_maxsolves=0	0.0341	0.0610	10050.28
2	alg=1, bar_murule=1, hessopt=1, multistart=1, ms_maxsolves=0	0.0341	0.0610	7202.76
73	alg=2, bar_murule=2, hessopt=5, multistart=0	0.0610	0.1026	8574.27
74	alg=2, bar_murule=2, hessopt=5, multistart=1, ms_maxsolves=0	0.0610	0.1026	13397.67
69	alg=2, bar_murule=2, hessopt=4, multistart=0	0.0626	0.1072	8574.99
70	alg=2, bar_murule=2, hessopt=4, multistart=1, ms_maxsolves=0	0.0626	0.1072	10429.50

Table 3.10 Bottom Results for the 15-Site Instance

No.	Solver Parameters	Lower Bound	Upper Bound	Total Solve Time(s)
24	alg=1, bar_murule=3, hessopt=6, lmsize=5, multistart=1, ms_maxsolves=10	0.3604	0.5978	10060.52
62	alg=2, bar_murule=1, hessopt=6, lmsize=5, multistart=1, ms_maxsolves=0	0.5890	0.8168	10060.09
64	alg=2, bar_murule=1, hessopt=6, lmsize=5, multistart=1, ms_maxsolves=10	0.5890	0.8168	10011.77
77	alg=2, bar_murule=2, hessopt=6, lmsize=5, multistart=0	12.6809	14.4208	8901.47
78	alg=2, bar_murule=2, hessopt=6, lmsize=5, multistart=1, ms_maxsolves=0	12.6809	14.4208	18627.25
79	alg=2, bar_murule=2, hessopt=6, lmsize=5, multistart=1, ms_maxsolves=1	12.6809	14.4208	6088.25
21	alg=1, bar_murule=3, hessopt=6, lmsize=5, multistart=0	2979.60	3040.78	9685.55
22	alg=1, bar_murule=3, hessopt=6, lmsize=5, multistart=1, ms_maxsolves=0	2979.60	3040.78	16608.36

No.	Solver Parameters	Lower Bound	Upper Bound	Total Solve Time(s)
23	alg=1, bar_murule=3, hessopt=6, lmsize=5, multistart=1, ms_maxsolves=1	2979.60	3040.78	9653.17
61	alg=2, bar_murule=1, hessopt=6, lmsize=5, multistart=0	7782.58	8046.52	6808.94
63	alg=2, bar_murule=1, hessopt=6, lmsize=5, multistart=1, ms_maxsolves=1	7782.58	8046.52	9698.45

From the results of the factorial experiments shown in the tables, one can have the following observations:

- Top results are achieved from experiments with parameter setting: $multistart = 1$ and $ms\_maxsolves = 0$. First, all the top results for the 9-site instance shown in Table 3.7 and all the top results for the 15-site instance shown in Table 3.9 have this parameter setting. Second, the average objective value with this parameter setting for the 9-site instance is 9.32, whereas it is 36.24 without this parameter setting.

- $hessopt = 6$ and $lmsize = 5$ produce undesirable results. All the bottom results for the 9-site instance shown in Table 3.8 and all the bottom results for the 15-site instance shown in Table 3.10 have this parameter setting. They also cause long solve time and sometimes cause time-out or an error of "Not Enough Memory". There are 3 timeout experiments for the 9-site instance. The parameter settings are:

$$\begin{cases} \text{alg=1, bar\_murule=1, hessopt=6, lmsize=5, multistart=1, ms\_maxsolves=0} \\ \text{alg=1, bar\_murule=3, hessopt=6, lmsize=5, multistart=1, ms\_maxsolves=0} \\ \text{alg=1, bar\_murule=4, hessopt=6, lmsize=5, multistart=0} \end{cases}$$

They all have this parameter setting "$hessopt = 6$ and $lmsize = 5$". For 15-site instance, the parameter setting "*alg=2, bar_murule=2, hessopt=6, lmsize=5, multistart=1, ms_maxsolves=10*" that causes the error—"Not enough memory" — also has "$hessopt = 6$ and $lmsize = 5$ ". Hence, considering the conclusion drawn from preliminary experiment that larger *lmsize* value causes solver timeout more often, we should avoiding specifying *hessopt* and *lmsize* values other than the default ones when solving the mathematical model.

- Generally, better results have higher cost (longer solve time). For example, for the 15-site instance, the average objective value of the experiments whose solve time ranks 1 to 20 is 0.1408 as opposed to 0.1211 for those experiments whose solve time rank from 21 to 40.

- When other parameters are set the same, Algorithm Interior/CG performs better than Interior/Direct for both problem instances but it takes longer. For example, considering all the experiments performed for 9-site instance, the results shown in Table 3.11 (only experiments with same parameters are compared) attests that observation.

Table 3.11 Comparison of Interior/Direct, Interior/CG Algorithms

Algorithm	Average Objective Value	Average Solve Time
1. Interior/Direct	1.632327	14.70891858
2. Interior/CG	1.59296	25.42268

- Small multiple start points like 10 elongate solve time but do not improve results. For example, experiment 3 and 4 for the 9-site instance, have the same parameter setting "*alg=1, bar_murule=1, hessopt=1, multistart=1*", except for *ms_maxsolves* . In experiment 3, $ms\_maxsolves = 1$. In experiment 4, $ms\_maxsolves = 10$. For these two experiments, the objective value is the same (1.6718), but experiment 4 takes 10 times longer time than experiment 3, 25.47 versus 247.29. This is not an individual case. The same is true for other experiments.

- Although the two problem instances have a large difference in size, traffic rate and other model parameters, the solver parameters have the similar impact on the results. Besides the fact that the previous conclusions apply to both instances, most parameter settings that produce top results shown in Table 3.7 (for the 9-site instance) and Table 3.9 (for the 15-site instance) are the same, although the exact order is different. The bottom results shown Table 3.8 and Table 3.10 show the same trend. The difference of exact order can be explained by the randomness of the traffic.

Chapter 4 Model Validation With Simulation

In this chapter, the mathematical programming model developed in the last chapter is validated using discrete-event simulation. The validation is to justify the assumptions that are made when the mathematical programming model is being built. Also, it is to increase one's confidence when the optimization results are applied to the design of real-world distributed PACS's.

The simulation model is first described. This includes the network model, traffic model, performance metrics, and the development of the simulator. Next, the experiments and results are presented. Results from optimization and simulation are compared.

4.1 Simulation Model Description

The major difference between the simulation model developed in this chapter and the mathematical programming model presented in the last chapter is the inclusion of a number of important components in the simulation model that are essential to real-world implementation of distributed PACS's. These components are application-layer, transport-layer, and network-layer protocols within end systems, network-layer protocol inside network routers, as well as buffer management and channel scheduling within network routers. They are left out from the mathematical programming model to simplify the modelling abstraction.

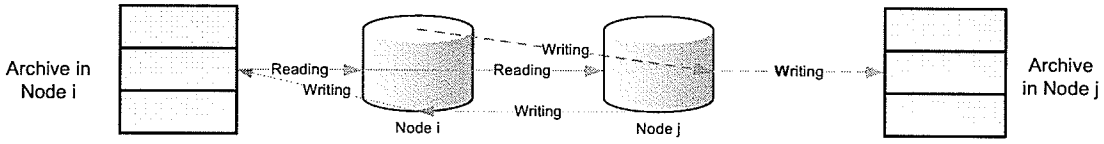
4.1.1 Network Model

The network model used consists of nodes and links. Each link connects two nodes, and consists of two channels, same as in Chapter 3. The topology and channel capacity are assumed to be given and fixed. Each network node relays traffic if it is an en route router. In addition, each node may generate traffic to be sent and terminate traffic to be received.

To model the archive delay, an archive node and a link are added for each node in the network as shown in Figure 4.1. An archive node is referred as an archive. The added link connects the added archive node to the network node where the archive is located. Such links are viewed as virtual network channels, thus the archive delay is modelled as the delay on a virtual network channel. It can be used as an approximation of real-world archive delays. This implementation is also consistent with that in the mathematical programming model (refer to Section 3.4).

In Figure 4.1, the solid line shows the direction of the traffic caused by one reading from Archive i to Node j. The "Reading" arrow between Archive i and Node i is a virtual network channel that is used to model archive delay within Archive i. The "Reading" arrow between Nodes i and j is a network channel which models the network delay. The dashed lines show two writing processes. A writing process is similar to a reading process, except in opposite direction. The detailed reading and writing processes are described in Section 3.2.1.

Figure 4.1 Archive Position in Data Transfers



The bandwidth of the virtual channel between a node and its archive is set to the capacity of the archive. The capacity of archives is part of the optimization result (variable z in the mathematical model). For simplicity, we assume that each channel other than the channels connecting archives has the same bandwidth. Note that if real circumstances warrant, setting different network channel bandwidth does not increase the complexity of the model.

Each network channel and each archive have a finite buffer. The buffer is to accommodate the packets waiting to be serviced by the network channels or archives. When creating channels, I use Droptail [19] queue management, which is commonly used on the Internet. Packets arriving at a full queue will be dropped. As for the channel scheduling algorithm, I used First Come First Serve (FCFS). This is also commonly used on the current Internet. Among network nodes, fixed shortest-path routing is used.

4.1.2 Traffic Model

The simulation simulates the transfers of individual studies or files from a source to a destination in a distributed PACS. Although PACS's do not use the FTP protocol to transfer data, but use DICOM instead, the transfer of the PACS data is very similar to FTP in that a large amount of data is transferred once a virtual connection is confirmed.

Moreover, like FTP, DICOM also uses TCP to communicate between systems [4]. Hence, the data transfers are simulated as FTP data transfers.

Each run of simulation takes in a traffic data file as an input. The traffic data file contains information about studies that are to travel through the distributed PACS being simulated. For each study, the information consists of its source node, destination node, size, and arrival time. Because the reading and writing data arrival processes have different source and destination, they are generated separately. As shown in Figure 4.1, an archive node is placed before the source node or after the destination node according to whether a data transfer is reading or writing traffic. For reading traffic, the archive node is placed before the source node to simulate the archive delay because the data is read from an archive first. For writing traffic, the data is written to the destination node and thus the archive node is placed after the destination node to simulate the archive writing delay.

In the simulation, arriving traffic of each route is generated according to a Poisson Process. Assume archives are identified separately from network nodes. The mean inter-arrival time t_{ij} on route ij is calculated by Equations (27) and (28):

$$t_{ij} = \frac{\eta}{\lambda_{ij}} \qquad i, j \in V \cup archives \tag{27}$$

$$\lambda_{ij} = \begin{cases} r_{ij} & \text{if i is the archive, j} \in V \cup archives \\ w_{ij} & \text{if j is the archive, i} \in V \cup archives \end{cases} \tag{28}$$

where η is the mean file size in Mb, and λ_{ij} is the data rate in Mbps on that route. $\frac{1}{t_{ij}}$ is the mean number of studies arrived per second on route ij. Equation (28) means λ_{ij} is

either the reading traffic (with rate r_{ij}) or the writing traffic(with rate w_{ij}) depending on where the archive is. i and j can not be both archives at the same time (refer to Section 3.1). For example, if there is reading traffic on route ij ($r_{ij} > 0$), i must be an archive and j is a node. if there is writing traffic on route i,j ($w_{ij}>0$), i must be a node and j must be an archive.

The size of each study to be transferred in the simulation is generated according to an exponential distribution.

4.1.3 Performance Metrics

The mean time to transfer one Mbit of data is used as the performance metric for each simulation run. It is obtained from simulation as follows.

Simulation output contains the following information for each the transfer of a study (a flow): the study size, the arrival time, the departure time, the source node, and the destination node. With these quantities, we can calculate the transfer time of each study by:

$$t_i = t_i^d - t_i^a \tag{29}$$

where t_i is the transfer time of study i, t_i^d and t_i^a are the arrival time and departure time of study i respectively.

The performance metric of each simulation run is calculated by:

$$t = \frac{\sum_{i \in N} t_i}{\sum_{i \in N} s_i} \tag{30}$$

where t is the mean time to transfer one Mbit of data. s_i is the size of Study i (in Mbits). N is the set of studies being transferred in each simulation run.

4.1.4 Simulation Development

The simulation performed in this research is packet-level simulation using NS2 [19]. The packet-level simulation simulates the lifecycle of network communication packets generated by the PACS, e.g., packets sending, queuing, routing, and receiving, etc.

There are plenty of software packages we can use to do packet-level network simulation. I choose NS2 because of its popularity, availability and, simplicity. NS2 is a discrete event simulator and has built-in support for the TCP/IP protocol stack such as FTP, TCP, UDP, and IP, which are used by real-world networks. TCP Sender/Receiver agents of NS2 are used to simulate source destination nodes. FTP components of NS2 are used for simulating transfer of studies.

As simulation plays an important role in this research, correctness of the simulation model is of concern. Hence, before I use the results of simulation to validate the optimization results, I verify the simulation model. Simulation model verification involves "ensuring that the computer program of the computerized model and its implementation are correct" [55]. Channel utilization is used to verify the simulation model. I compare the queuing theoretical utilization with the utilization calculated using the data collected from simulation. The theoretical utilization is equal to the traffic rate in

Mbps on each channel divided by the channel bandwidth in Mbps. The traffic rate on each channel is equal to the summation of the traffic rate of the flows that go through the channel. Equation (31) shows the formula:

$$util_{st} = \frac{\sum_{i,j \in V} R_{ijst} \lambda_{ij}}{B_{st}} \qquad \forall st \in L \tag{31}$$

where $util_{st}$ is the theoretical utilization on channel st. Refer to Chapter 3 for the meaning of the other notations.

The other utilization is calculated by using the statistics gathered during the simulation running process. NS2 provides an object called queue-monitor that collects statistics about the queue on each channel [56]. The throughput is one of the statistics. The utilization of a channel is the throughput of that channel divided by the channel bandwidth.

Table A.1 in Appendix A shows one set of results for a 9-node network. Results from simulation and from queuing theory fundamental results match well. Further experiments with different parameters and other problem instance show the same trend.

As this project needs many types of data manipulation to generate the data input for the solver and the simulation programs, I use Python and MATLAB to do the input data formatting.

To deal with the variation in simulation results and to improve the confidence in results, multiple replications are performed for each experiment. Considering that the simulation is time-consuming, the number of replications is not too large, 12 to be exact. Sample

means of the replication results are collected and used for the analysis of the results. 95% confidence interval is computed in order to see the variations of the results.

4.2 Experiments and Results

This section describes the experiments performed for solving the mathematical programming model for different input parameters, the corresponding simulation experiments, and their results.

4.2.1 Overview of Experiments

Besides the solver parameter tuning experiments of solving the mathematical model I described in Chapter 3, I did other three sets of experiments with regard to simulation. They are:

- Experiments of simulation for different solver parameter settings for which I have solved the mathematical programming model during solver parameter tuning. This set of experiments is to see how the results of solving the mathematical model compare with the simulation results under different solver parameter settings. When choosing parameter settings for which I have done solver parameter tuning to do simulation, I choose those that produce the best, average, worst performance metric. I then observe the simulation results to see if there is the same type of difference in the performance metric.

- Experiments with different budget.

- Experiments with different network bandwidth.

The main purpose of the last two sets of experiments is to find out how the two main system factors impact the performance metrics. When designing a distributed PACS, the two main factors that concern the designers are:

- Bandwidth in Mbps of the channels in the network (B in Table 3.1).
- Budget or Total Capacity (C in Table 3.1).

They correspond to parameters B and C in Table 3.1 respectively. When I vary the two system factors described above, I fix the values of the solver parameter settings. On the other hand, when I perform simulation for different solver parameter settings, the values of the main factors are fixed.

Another purpose of all the experiments is to find out if the results from simulation experiments validate the results from solving the mathematical programming model. Hence, the experiments of solving the mathematical programming model and the simulation experiments are performed in tandem. For each set of parameters for which I solve the mathematical programming model, I also run a simulation. To make the two sets of results comparable, simulations use the same parameters as the mathematical programming model. The same parameters are the bandwidth, number of sites, network topology and routing. The results from solving the mathematical programming model, the placement of studies and the capacities of archives, will also be used as part of the parameters for simulations.

The results of solving the mathematical programming model with different factors and different solver parameters settings will be compared with the results from the

corresponding simulation experiments. The results to be compared include the performance metrics and how the change of the factors affects the performance metrics.

We expect to see the factors affect the results of the two types of experiments the same way. When one factor changes, the two sets of values of the performance metric from the two types of experiments should both increase and decrease in the same direction and in similar amount. They also should change according to common sense. For example, when the budget or the bandwidth increases/decreases, we expect the objective value or the performance metrics decrease/increase. We also expect to see that for different performance as a result of different solver parameter settings, simulation results show the same difference in performance metrics.

4.2.2 Input Parameters

The simulation uses the same two problem instances as does the mathematical programming model. The network parameters of the two problem instances are described in Section 3.6.1.

The reading and writing traffic rate on each route from the solution to the mathematical programming model are plugged in Equations (27) and (28) to calculate the mean inter-arrival time of the files to be transferred in the simulation.

Values of the two main system factors are chosen in a way that the change of the value is effective on the performance metric. When the network is more congested, it is more effective to change the factors. Hence, I choose the values of parameters that make the highest network channel utilization close to 90%. To get the experiment levels that make

the network channel utilization high, I start experimenting with small levels that produce infeasible solutions. For example, when C is set 30, and B 0.5, the Solver reports that it can not find feasible solutions. I then relax the levels of the two factors until feasible solutions are found.

Because the parameters values that achieve the same level of network congestion are different for the two problem instances, the selected levels of the factors are different. They are listed in Table 4.1 and Table 4.2 respectively.

- 9-Site Instance

Table 4.1 Factors and Levels for the 9-Site Instance

Factors	**Levels**
Budget	40, 50, 60, 70, 80
Bandwidth (Mbps)	1, 1.5, 2, 2.5, 3

There are 5 levels for each factor. With these levels, there are a total of 50 experiments, 25 on solving the mathematical programming model and 25 on simulation (each experiment has 12 replications)

- 15-Site Instance

Table 4.2 Factors and Levels for the 15-Site Instance

Factors	**Levels**
Budget	1500, 1750, 2000, 2250, 2500

Bandwidth(Mbps)	100

There are 5 levels for budget and one level for bandwidth. With these levels, there are a total of 10 experiments, 5 on solving the mathematical model and 5 on simulation (each experiment has 12 replications). I only choose one level for bandwidth in order to save time on experimenting. Also, the budget is more important to PACS designers.

When performing the above experiments, I select the following parameters for the KNITRO solver because experiments performed for tuning KNITRO solver parameters show that it produces good objective value in a relatively short time:

$alg = 1;\ bar\_murule = 4;\ hessopt = 1;\ multistart = 0$

The mean file size is set to 25MB, which is given in [53] by Otukile, for both problem instances. The arrival rate of studies on route ij (α_{ij}) is calculated by Equation (32). Equation (32) is the result of combining Equations (27) and (28).

$$\alpha_{ij} = \frac{25MB * 8}{\lambda_{ij}} \qquad i, j \in V \cup archives \tag{32}$$

Because the unit of λ_{ij} is Mbps (refer to Section 3.3), the size is converted to bits. The number of studies to be transferred in each run of the simulation is 1000.

4.2.3 Comparison of Optimization and Simulation Results

After each set of mathematical model solving experiment and simulation experiment with the same parameters, I compare the performance bounds from solving the mathematical model and performance metric calculated from simulation results.

Figure 4.2 and Figure 4.3 show the results of experiments performed for different solver parameter settings for the 9-site instance and the 15-site instance respectively. Three parameter settings are experimented. They produce one of the best, average, and one of the worst objective function values respectively during optimization. These settings are shown as index number on the X-axis in the two figures. Their corresponding parameter values are described in Table 4.3. Each figure has three lines showing the 2 bounds (lower and upper bound) of the performance metric from solving the mathematical model and the performance metric from the corresponding simulation. In the two figures, we can see the simulation performance changes accordingly as the performance from the analytical model changes.

Table 4.3 Parameter Setting Index

Index	**9-Site Instance**	**15-Site Instance**
1	alg=1, bar_murule=6, hessopt=1, multistart=1, ms_maxsolves=0	alg=1, bar_murule=2, hessopt=1, multistart=1, ms_maxsolves=0
2	alg=2, bar_murule=1, hessopt=1, multistart=0	alg=2, bar_murule=2, hessopt=4, multistart=0
3	alg=2, bar_murule=1, hessopt=4, multistart=0	alg=1, bar_murule=1, hessopt=1, multistart=1, ms_maxsolves=1

Figure 4.2 Comparison of the Performance Metric for the 9-Site Instance

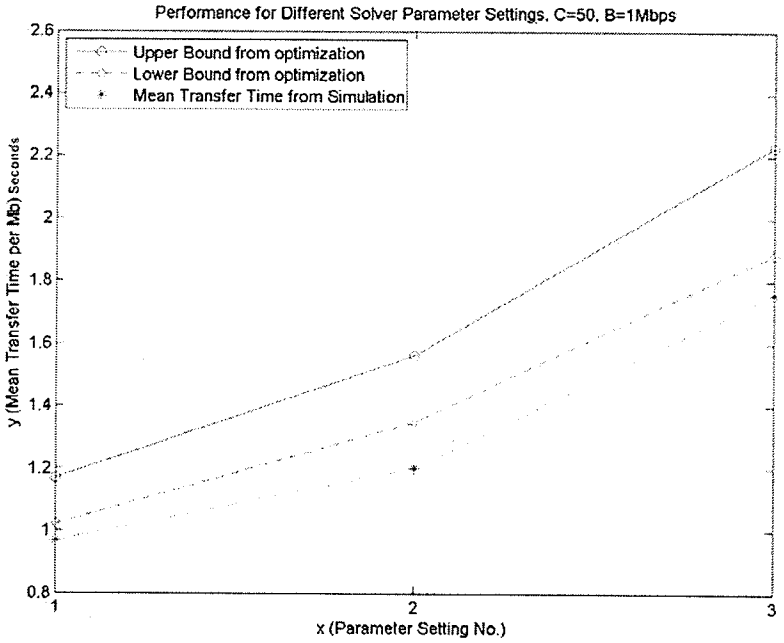
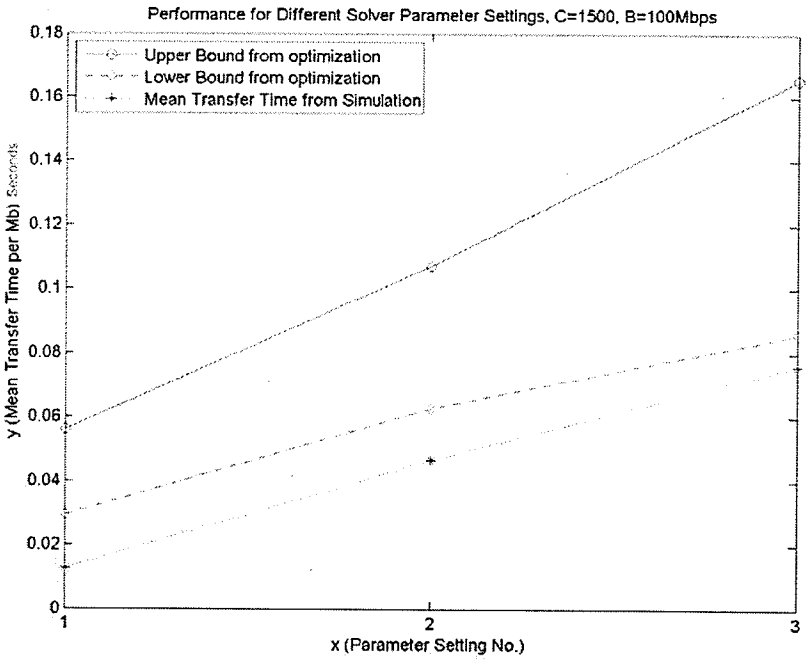


Figure 4.3 Comparison of the Performance Metric for the 15-Site Instance



Also, in experiments performed for other solver parameter tuning experiments, the comparison of the two sets of performance metrics from optimization results and simulation results exhibit the same characteristics.

Furthermore, for the experiments performed for different factor levels for the two problem instances, the previous observation of the performance metrics still holds true. No matter which factor is changed, (either the budget or the channel bandwidth), both performance metrics increase or decrease in the same direction and in similar amount. The change of the bandwidth and the budget affects the simulation results the same way as it does the optimization results. Ensuing Subsections 4.2.4 and 4.2.5 will show this in detail.

Therefore, the simulation model behaves the same way as does the mathematical programming model for all experiments performed for different factor levels and solver parameter settings. This increases the users' confidence in the mathematical programming model.

All experiments including those shown here consistently show that if the solution of the mathematical programming model produces a placement of archives whose performance metric is better than that of another placement, the simulation results corroborate.

We also notice that the average transfer time from simulation is smaller than the lower bound of the mean transfer time from the optimization results. There are a number of possible reasons for this phenomenon:

First, the assumptions of the mathematical programming model are different than those of the simulation in the following aspects:

> "Mathematical programming modelling assumes a fluid model; the flow is transferred through the network as a continuous stream; no storing of data in links or queues. When a flow starts, it is immediately received at the destination at the same rate as it is being sent." Page 4 in [39]

Second, the mathematical programming model and simulation assume a different fairness scheme. The mathematical programming model assumes the balanced fairness scheme whereas the simulation uses TCP.

Third, as is mentioned in the beginning of Section 4.1, the simulation model models in much more detail the PACS than does the mathematical programming model.

We conclude that the performance metric from the mathematical programming model can be a conservative estimate of the actual performance.

4.2.4 The Effect of the Budget

Understandably, designers of distributed PACS usually are concerned with the trade-off between the budget and the performance. Hence, experiments are performed to explore the effect of the budget on performance.

The experiment objective is to see how the varying of the budget changes the results and to see if varying the budget affects the results of the mathematical programming model as those of the simulation. Also, to see if better placements produced by the mathematical programming model as a result of increasing budget indeed have better performance shown in simulation results.

For the experiments, I fix the bandwidth value and change the budget value. The detailed factors and levels are described in Section 4.2.2.

Figure 4.4 shows one set of results of changing the budget for the 9-site instance when the bandwidth of all channels is set to 1Mbps. Figure 4.5 shows another set of results of changing the budget for the 15-site instance when the bandwidth of all channels is set to 100Mbps. It is clearly shown in the two figures that the trend of the change of the performance metric from the simulation results matches very well with that from the analytical results. Also, for both problem instances, increasing budget improves the results of the mathematical programming model and simulation results. The results meet our expectation. Another useful observation is to find the point where the decrease of the mean transfer time levels off. This point is where the investment in the bandwidth or the increase in the budget starts to become less effective. For example, in Figure 4.4, the increase of the budget when it is greater than 60 becomes less advantageous in terms of improving system performance.

Figure 4.4 Effect of Budget on the 9-Site Instance

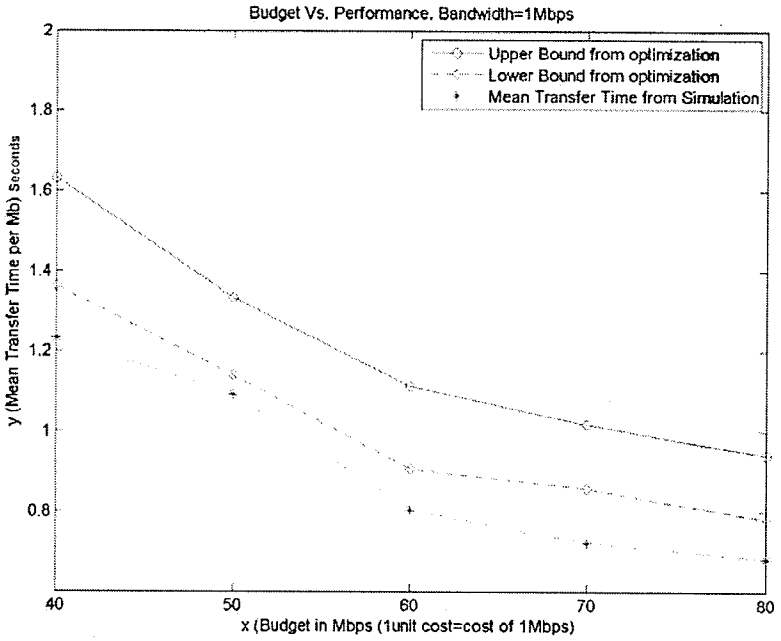
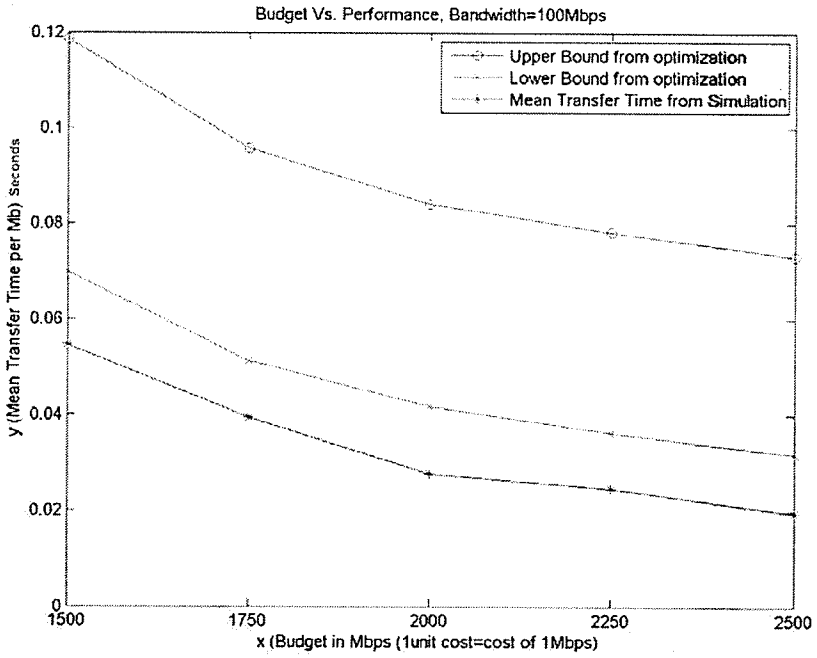


Figure 4.5 Effect of Budget on the 15-Site Instance



4.2.5 The Effect of Bandwidth

When a network like a PACS spans across a health region, provisioning network channels are also an important designing decision. This is because the unit cost for the bandwidth of network connections over long distance is usually high. The designers would like to have the knowledge of the effect of different network bandwidth on the performance metric before they make the decision. Moreover, to further explore how the simulation results compare to the results from solving the mathematical programming model under a different scenario, I perform more experiments with varying channel bandwidth of the network.

The experiment objective is to see how the varying of the bandwidth changes the results and to see if the changing of the bandwidth affects the results of the mathematical programming model as those of the simulation. Also, to see if better placements produced by the mathematical programming model as a result of increasing bandwidth indeed have better performance shown in the simulation results.

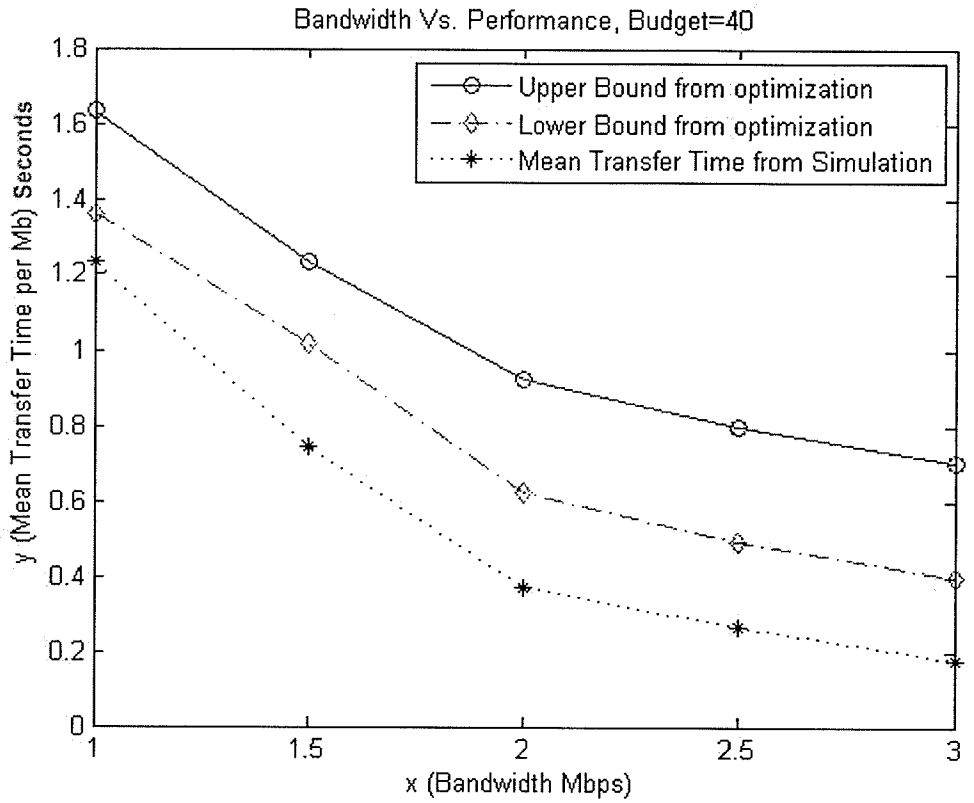
For the experiments, I fix the budget value and change the bandwidth value. The detailed factors and levels are described in Section 4.2.2.

Figure 4.6 shows one set of the results of changing the bandwidth for the 9-site instance when the budget is fixed to 40. It is shown in the figure that increasing the bandwidth improves the performance metric of the mathematical programming model and the simulation. These results meet our expectation too. The point where the bandwidth is 2Mbps can be regarded as the level-off point.

The experiments shown in the figures in this section are only a small part of experiments I have performed when exploring how the two sets of performance metric compare and how to best use the simulation model to validate the mathematical programming model. However, these experiments provide a bird's eye view of how the simulation results validate the results from solving the mathematical programming model. The other omitted experiments do not conflict the experiments presented here.

Based on all the experiments performed, I conclude that (i) the mathematical programming model produces good placements. The quality of these placements is verified by the simulation. The average transfer time from the optimization is slightly higher than from the simulation due to more realistic conditions modeled in the simulation, thus the results from optimization can be considered conservative estimates on the actual performance. (ii) The change of budget and bandwidth results in the desired effect and is consistent with their effect on results from solving the mathematical programming model. Therefore, the optimization model and the framework developed in this work can be considered an attractive alternative to designers of distributed PACS's.

Figure 4.6 Effect of Bandwidth on the 9-Site Instance



Although hitherto all the experiments have been designed for distributed PACS's, I believe the methodology can be used to other problems with similar characteristics.

Chapter 5 Conclusions

Technology advances in the radiology department in the health industry increase the amount of digital medical images produced and thus the demand to share the image data due to the mobility of patients, modality sharing and consultation between physicians. This trend demands an architectural shift from the current centralized mode. However, designing a distributed PACS is a complicated decision which requires one to decide the best locations, number, and capacity of digital image archives under a certain budget. I have not found research work that has been done to solve this emerging problem.

In this project, I have tackled this problem through two major steps. First, an existing mathematical programming model is introduced for the problem, then modified and solved using an existing solver. While the model is being presented, performance bounds are introduced to measure the performance of a distributed PACS. While solving the model using an existing solver, I have explored the parameter settings of the solver to produce significantly better results than using the default parameters settings. Second, I have carried out packet-level simulations whose results give evidence of the validity of the analytical model. Before utilizing the simulation model, I verified it by comparing the utilization based on data collected from simulation with the utilization based on queuing theory. The experiment results show that the analytical model is valid and that the methodology developed can help designers to choose the location, number and capacity of archives to get the best performance under a certain budget. In addition, it can help to find the best amount of investment that could bring the most performance improvement

when designing a distributed PACS. The detailed contributions of this thesis project are listed in next section.

5.1 Contributions

The contributions of this thesis project are as follows:

- Utilizing a model to design a distributed PACS or other systems with similar characteristics. The model aims to find the best location, capacity of PACS archives for a distributed PACS to optimize the performance under a certain budget. Two problem instances are constructed and solved to illustrate the usefulness of the model.

- Providing the tool that decision makers can use to find out:

    a. Given a specific budget for a PACS network model, what is the best performance that could be achieved.

    b. Where to place the archives and how to provision them to achieve best performance for a given budget.

    c. If the current best performance from a certain budget is unsatisfactory, how performance will improve as a result of increasing the budget and where the level-off point (where the investment increase becomes less rewarding) is to be found. By solving the problem instances of different budget constraints, decision makers can find out when performance can be

d. Where the bottlenecks are, be it archives or network channels, the tool can locate them and show their utilization. From this information, they can pinpoint the performance problem and prioritize the investment.

- Designing, implementing, and verifying a simulation model to evaluate and verify the optimization results. Simulations verify if the assumptions made during creating the mathematical programming model are appropriate. Simulation will give more confidence to decision-makers when they make use of the optimization results to design a distributed PACS archive system. Simulate the mathematical programming model at packet level using NS2.

- Tuning the parameters of the KNITRO solver, presenting the results showing the impact of different combinations of important solver parameters, and providing insight of how to use the solver parameters.

- Providing methods to analyze the performance of an existing distributed PACS network.

5.2 Future Work

Possible future work to improve or expand this research includes:

- Finding a new approach to deriving a more accurate network file transfer performance metric than the bound on average delay that is used in this research.

- Venturing further research on other types of simulations besides packet-level simulation and find out their advantages and disadvantages compared to the simulation that I have done. Flow-level simulation is one possible direction.

- Implementing a more accurate computer system performance model to simulate an archive in a distributed PACS system.

Appendix A Results of Simulation Verification by Utilization

Table A.1 Comparison of Two Types of Utilization

Bandwidth is set to 2Mbps, budget 50

Channel Start	Channel End	Theoretical Utilization	Utilization From Simulation	Difference
1	2	70.00%	71.59%	2.26%
1	3	60.17%	59.86%	0.52%
1	6	35.19%	35.05%	0.39%
1	7	42.37%	41.98%	0.93%
1	1	74.17%	73.81%	0.48%
2	1	63.15%	63.16%	0.01%
2	3	49.25%	50.57%	2.69%
2	4	32.02%	31.79%	0.71%
2	8	26.66%	27.40%	2.78%
2	2	66.85%	66.54%	0.47%
3	1	53.00%	54.06%	2.00%
3	2	52.72%	53.88%	2.20%
3	5	50.39%	48.95%	2.86%

Channel Start	Channel End	Theoretical Utilization	Utilization From Simulation	Difference
3	9	29.43%	28.57%	2.92%
3	3	50.59%	50.58%	0.02%
4	2	33.92%	34.19%	0.80%
4	8	4.35%	4.26%	1.98%
4	4	30.23%	29.74%	1.62%
5	3	45.96%	45.76%	0.43%
5	7	22.13%	21.80%	1.51%
5	5	45.10%	43.84%	2.80%
6	1	34.46%	33.48%	2.84%
6	9	4.47%	4.59%	2.69%
6	6	31.55%	32.22%	2.12%
7	1	37.37%	37.36%	0.03%
7	5	24.34%	24.04%	1.23%
7	7	45.54%	45.23%	0.68%
8	2	25.50%	25.07%	1.67%
8	4	2.03%	1.98%	2.26%
8	8	31.68%	31.55%	0.41%

Channel Start	Channel End	Theoretical Utilization	Utilization From Simulation	Difference
9	3	28.69%	28.45%	0.85%
9	6	2.66%	2.71%	1.70%
9	9	32.88%	32.42%	1.40%

References

[1] O. R. L. Sheng and H.-M. C. Garcia. The design of medical image databases: A distributed approach, In *Proceedings of Ninth Annual International Phoenix Conference on Computers and Communications* , pages 2808-2895, March 21-23 1990.

[2] Cisco wide area application services version 4.0 optimizations for PACS and digital image storage, Cisco Systems Inc., http://www.cisco.com/en/US/products/ps6870/products_white_paper0900aecd8051d5d8.shtml, March 2007.

[3] J. Bellavance. When PACS Pushes the Limit - Image Storage: The Basics, http://www.healthimaging.com/content/view/6415/68/May 1, 2007.

[4] H. K. Huang. PACS and imaging informatics: basic principles and applications, 2nd ed., Wiley-Liss, 2004.

[5] J. H. T. Keith J. Dreyer, David S. Hirschorn and Amit Mehta. PACS, 2nd ed., Springer New York, 2006.

[6] T. Bonald and A. Proutière. On performance bounds for balanced fairness, *Performance Evaluation,* vol. 55, no. 1-2, pages 25-50, January 2004.

[7] S. R. Mogatala. Web cache location and provisioning for a regional internet service provider, Master's Thesis of Engineering, in *Department of Electrical and Computer Engineering* Winnipeg, Manitoba: University of Manitoba, 2005.

[8] Wikipedia. Digital Imaging and Communications in Medicine, http://en.wikipedia.org/wiki/Digital_Imaging_and_Communications_in_Medicine

[9] DICOM strategic document, 7.2 ed: National Electrical Manufacturers Association, 2007.

[10] H. Vagelis, J. C. Peter, P. Nagarajan, D. Yi, A. W. Jeffrey, and P. B. Redmond. A flexible approach for electronic medical records exchange, in *Proceedings of the international workshop on Healthcare information and knowledge management* Arlington, Virginia, USA: ACM, 2006.

[11] E. Marco, A. Thomas, J, R. rg, D. Asuman, and B. L. Gokce. A survey and analysis of Electronic Healthcare Record standards, *ACM Comput. Surv.,* vol. 37, no. 4, pages 277-315, 2005.

[12] D. Bandon, C. Lovis, A. Geissbhler, and J.-P. Valle. Enterprise-wide PACS: Beyond radiology, an architecture to manage all medical images, *Academic Radiology,* vol. 12, no. 8, pages 1000-1009, 2005.

[13] H. K. Huang. Enterprise PACS and image distribution, *Computerized Medical Imaging and Graphics,* vol. 27, no. 2-3, pages 241-253, 2003.

[14] M. Tsiknakis, D. G. Katehakis, and S. C. Orphanoudakis. Intelligent image management in a distributed PACS and telemedicine environment, *Communications Magazine, IEEE,* vol. 34, no. 7, pages 36-45, July 1996 July 1996

[15] J. Diamond, P. Zhou, S. Camorlinga, M. Toulouse, and E. Liu. Archive planning for a metropolitan PACS network, In *Proceedings of The 22nd Meeting of the Society for Computer Applications in Radiology Conference (SCAR 2005)*, June 2005.

[16] G. R. Lawrence, G. A. Marin, and S. E. Naron. Simulation of a hospital picture archiving and control system (PACS), in *Proceedings of the 17th conference on Winter simulation* San Francisco, California, United States: ACM, 1985.

[17] T. Chiotis, T. Karounos, and B. Maglaris. Performance evaluation and network management of picture archiving and communication systems-PACS, In *Proceedings of the 4th International Conference on Advances in Communication and Control*, page 213, 1993.

[18] H. K. Huang, Z. Aifeng, L. Brent, Z. Zheng, D. Jorge, K. Nelson, and L. W. C. Chan. Data grid for large-scale medical image archive and analysis, in

Proceedings of the 13th annual ACM international conference on Multimedia Hilton, Singapore: ACM, 2005.

[19] Network Simulator, http://www.isi.edu/nsnam/ns/

[20] M. L. Averill and G. M. Michael. Simulation of communications networks, In *Proceedings of the 28th Conference on Winter simulation*, IEEE Computer Society, 1996.

[21] L. Kleinrock. Queueing systems, Wiley, 1975.

[22] B. Liu, D. R. Figueiredo, Y. Guo, J. F. Kurose, and D. F. Towsley. A Study of Networks Simulation Efficiency: Fluid Simulation vs. Packet-level Simulation, In *Proceedings of IEEE, INFOCOM*, pages 1244-1253, 2001.

[23] K. Cameron, S. Rob, W. Carey, and U. Brian. Hybrid packet/fluid flow network simulation, In *Proceedings of The Seventeenth Workshop on Parallel and Distributed Simulation*, pages 143-152, IEEE Computer Society, 2003.

[24] T. Yung, J. Martin, M. Takai, and R. Bagrodia. Integration of fluid-based analytical model with packet-level simulation for analysis of computer networks, 2001.

[25] I. S. Monirul, F. R. George, and L. Wenke. Comparative Study between Analytical Models and Packet-Level Worm Simulations, in *Proceedings of the 19th Workshop on Principles of Advanced and Distributed Simulation*: IEEE Computer Society, 2005.

[26] S. Camorlinga and B. Schofield. Impact of a workflow engaged network on radiology image transfers across a metro network, St. Boniface General Hospital Research Centre, Winnipeg, Manitoba, Submitted to IEEE Transactions on IT in Biomedicine 2005.

[27] W. Tu, C. J. Sreenan, and W. Ji. Worst-Case Delay Control in Multigroup Overlay Networks, *Parallel and Distributed Systems, IEEE Transactions on,* vol. 18, no. 10, pages 1407 - 1419, Oct. 2007.

[28] W. M. Moh, Y.-J. C. Zhang, and T. I. Moh. Delay performance evaluation of high speed protocols for multimedia communications, In *Proceedings of Computer Communications and Networks, Fourth International Conference on*, pages 352 - 355, 1995.

[29] J. F. Kurose and K. W. Ross. Computer networking: a top-down approach featuring the Internet, 2nd ed., Addison-Wesley, 2001.

[30] H. C. Gromoll, A. L. Puha, and R. J. Williams. The fluid limit of a heavily loaded processor sharing queue, *Annuals of Applied Probability,* vol. 12, no. 3, pages 797-859, 2002

[31] N. Chen; and S. Jordan. Throughput in processor-sharing queues, *Automatic Control, IEEE Transactions on,* vol. 52, no. 2, pages 299 - 305, Feb. 2007.

[32] A. M. Law and W. D. Kelton. Simulation modeling and analysis, 2nd ed., McGraw-Hill, 1991.

[33] K. Park and W. Willinger. Self-similar network traffic and performance evaluation, Wiley, 2000.

[34] T. Bonald and A. Proutière. Insensitive bandwidth sharing, In *Proceedings of IEEE GLOBECOM '02*, pages 2659-2663, 2002.

[35] S. B. Fred, T. Bonald, A. Proutiere, R. G, gni, and J. W. Roberts. Statistical bandwidth sharing: a study of congestion at flow level, In *Proceedings of the 2001 Conference on Applications, technologies, architectures, and protocols for computer communications*, ACM Press, 2001.

[36] T. Bonald and A. Proutière. Insensitive bandwidth sharing in data networks, *Queueing Systems: Theory and Applications Archive,* vol. 44, no. 1, pages 69 - 100, May 2003.

[37] T. Bonald and A. Proutière. Insensitivity in processor-sharing networks, *Performance Evaluation,* vol. 49, no. pages 193-209, 2002.

[38] T. Bonald, A. Proutière, G. Régnié, and J. W. Roberts. Insensitivity results in statistical bandwidth sharing, In *Proceedings of ITC 17 Conference*, 2001.

[39] V. Timonen. Simulation studies on performance of balanced fairness, Master's Thesis of Engineering, in *Department of Engineering Physics and Mathematics*. vol. Master of Science: Helsinki University of Technology, 2003.

[40] L. Massouliè and J. Roberts. Bandwidth sharing: Objectives and algorithms, In *Proceedings of IEEE INFOCOM*, pages 1395-1403, March 21-25 1999.

[41] F. Kelly. Charging and rate control for elastic traffic, In *Proceedings of European Transactions on Telecommunications*, pages 33-37, January 1997.

[42] L. Massouliè and J. W. Roberts. Bandwidth sharing and admission control for elastic traffic, *Telecommunication Systems,* vol. 15, no. 1-2, pages 185-201, 2000.

[43] B. Thomas and M. Laurent. Impact of fairness on Internet performance, In *Proceedings of the 2001 ACM SIGMETRICS international conference on Measurement and modeling of computer systems*, ACM Press, 2001.

[44] L. Massoulie and J. Roberts. Bandwidth sharing: Objectives and algorithms, In *Proceedings of Proceedings of the INFOCOM*, pages 1395-1403, 1999.

[45] J. Mo and J. Walrand. Fair end-to-end window-based congestion control, *IEEE/ACM Transactions on Networking,* vol. 8, no. 5, pages 556-567, 2000.

[46] KNITRO® --nonlinear optimization problems (NLP) solver: Ziena Optimization http://www.ziena.com/knitro.htm.

[47] M. M. J. Czyzyk, and J. Mor. The NEOS Server, *IEEE Journal on Computational Science and Engineering,* vol. 5, no. pages 68-75, 1998.

[48] R. H. Byrd, J. Nocedal, and R. A. Waltz. KNITRO: an integrated package for nonlinear optimization, In *Proceedings of Large-Scale Nonlinear Optimization*, pages 35-59, 2006.

[49] D. Steiger and R. Sharda. LP modeling languages for personal computers: A comparison., *Annals of Operations Research,* vol. 43, no. 3, pages 195-216, March, 1993 1993.

[50] R. Fourer, D. M. Gay, and B. W. Kernighan. AMPL a modeling language for mathematical programming, 2nd ed., Duxbury Press, 2004.

[51] R. A. Waltz and T. D. Plantenga. KNITRO 5.1 user's manual: Ziena Optimization Inc., Northwestern University, March 2007.

[52] P. C. Steven, L. G. Bruce, C. R. George, and A. W. Edward. Using experimental design to find effective parameter settings for heuristics, vol. 7, no. 1, pages 77-97, 2001.

[53] M. Otukile. Simulation and modeling of a 10 GB/s metropolitan area network for radiology, Master's Thesis of Engineering, in *Department of Electrical and Computer Engineering* Winnipeg, Manitoba: University of Manitoba, 2004.

[54] S. Camorlinga and B. Schofield. Modeling of workflow-engaged networks on radiology transfers across a metro network, *IEEE Transactions on Information Technology in Biomedicine,* vol. 10, no. 2, pages 275-281, April 2006.

[55] G. S. Robert. Verification and validation of simulation models, In *Proceedings of the 37th conference on Winter simulation*, Winter Simulation Conference, 2005.

[56] K. Fall and K. Varadhan. The NS manual: UC Berkeley, LBL, USC/ISI, and Xerox PARC, 2007.