A CLUSTERING-BASED SELECTIVE PROBING FRAMEWORK TO SUPPORT INTERNET QUALITY OF SERVICE ROUTING

by

Nattaphol Jariyakul

B.E. in Computer Engineering, Kasetsart University, 2000

Submitted to the Graduate Faculty of

Information Sciences in partial fulfillment

of the requirements for the degree of

Master of Science in Telecommunications

University of Pittsburgh

2004

UNIVERSITY OF PITTSBURGH

SCHOOL OF INFORMATION SCIENCES

This thesis was presented

by

Nattaphol Jariyakul

It was defended on

December 9, 2004

and approved by

Dr. Prashant Krishnamurthy, Assistant Professor

Dr. David Tipper, Associate Professor

Dr. Taieb Znati, Professor
Thesis Advisor

A Clustering-based Selective Probing Framework to Support Internet Quality of Service Routing

Nattaphol Jariyakul, MST

University of Pittsburgh, 2004

The advent of the multimedia applications has triggered widespread interest in QoS supports. Two Internet-based QoS frameworks have been proposed: Integrated Services (IntServ) and Differentiated Services (DiffServ). IntServ supports service guarantees on a per-flow basis. The framework, however, is not scalable due to the fact that routers have to maintain a large amount of state information for each supported flow. DiffServ was proposed as an alternate solution to address the lack of scalability of the IntServ framework. DiffServ uses class-based service differentiation to achieve aggregate support for QoS requirements. This approach eliminates the need to maintain per-flow states on a hop-by-hop basis and reduces considerably the overhead routers incur in forwarding traffic.

Both IntServ and DiffServ frameworks focus on packet scheduling. As such, they decouple routing from QoS provisioning. This typically results in inefficient routes, thereby limiting the ability of the network to support QoS requirements and to manage resources efficiently. The goal of this thesis is to address this shortcoming. We propose a scalable QoS routing framework to identify and select paths that are very likely to meet the QoS requirements of the underlying applications. The tenet of our approach is based on seamlessly integrating routing into the DiffServ framework to extend its ability to support QoS requirements. Scalability is achieved using selective probing and clustering to reduce signaling and routers overhead.

The major contributions of this thesis are as follows: First, we propose a scalable routing architecture that supports QoS requirements. The architecture seamlessly integrates the QoS traffic requirements of the underlying applications into a DiffServ framework. Second, we propose a new delay-based clustering method, referred to as *d*-median. The proposed clustering method groups Internet nodes into clusters, whereby nodes in the same cluster exhibit equivalent delay characteristics. Each cluster is represented by anchor node. Anchors use selective probing to estimate QoS parameters and select appropriate paths for traffic forwarding.

A thorough study to evaluate the performance of the proposed *d*-median clustering algorithm is conducted. The results of the study show that, for power-law graphs such as the Internet, the *d*-median clustering based approach outperforms the set covering method commonly proposed in the literature. The study shows that the widely used clustering methods, such as set covering or *k*-median, are inadequate to capture the balance between cluster sizes and the number of clusters. The results of the study also show that the proposed clustering method, applied to power-law graphs, is robust to changes in size and delay distribution of the network. Finally, the results suggest that the *delay bound* input parameter of the *d*-median scheme should be no less than 1 and no more than 4 times of the average delay per one hop of the network. This is mostly due to the weak hierarchy of the Internet resulting from its power-law structure and the prevalence of the small-world property.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

PREFACE

I would like to thank my family and Ploy for the continuous support and encouragement. I would also like to thank Chatree Sangpachatanaruk for many useful discussions and suggestions.

1.0 INTRODUCTION

The purpose of this chapter is to give the reader the background and the concept of the thesis. We begin this chapter by introducing the characteristics of the emerging class of multimedia applications and their diverse requirements on the networking resources. This invokes the demand of evolving the Internet beyond its one-class, best-effort service. We then state the research problem of this thesis and describe briefly the approach used to address this problem. Later, we provide some important results. Finally, we describe the organization of the rest of the thesis.

1.1 Background

The Internet is now unmistakably the most prominent communication infrastructure, carrying an ever broadening range of applications and protocols. The traditional best-effort service of the Internet is inadequate in supporting diverse characteristics and different *Quality-of-Service* (QoS) requirements of the new emerging applications. These applications, loosely referred to as multimedia applications, exchange information often characterized by real-time performance specifications and a wide range of bandwidth requirements. Examples of such applications include high-definition television, distance learning, remote video conferencing, video-on-demand services, retail sales and computer-based interactive game play. Multimedia support encompasses different requirements at different levels ranging from storage, retrieval and processing of multimedia information to its transmission and display. Depending upon the application and media type, such requirements may involve stringent temporal constraints.

Different multimedia applications are sensitive to different factors and possess a variety of service constraints, including bandwidth, delay bounds and loss bounds.

The recent advances in communication and processing system designs have made it possible to support such diverse requirements in a single, integrated environment [STAL02], forming a heterogeneous network. However, delivering certain level of QoS for every application simultaneously is difficult, taking into account the different in characteristic of the traffic. For example, the burstiness traffic may harm the low jitter requirement of some multimedia traffic at the aggregate points. The end-to-end flow control and congestion control of the higher layer protocols, e.g. the TCP's, are insufficient to cope with the problem. Therefore, the network must play an important role in delivering different QoS requirements to each class of traffic through variety of techniques, including admission control, constraint based routing, scheduling, queuing discipline, and discard policy. It's important to note that the service differentiation and QoS are essential even when the network does not reach the congestion point.

While IP-based Internet network had difficulties in providing such diverse QoS requirements, there was an effort, through the collaboration of the vendors and the service providers, in developing an entirely new network scheme, which aims to support vast variety of applications, including the emerging multimedia applications. This network, developed as part of the work on broadband ISDN in early 90's, is called *Asynchronous Transfer Mode* (ATM), featuring connection-oriented logical connections, fixed-size cells, and service categories. ATM exercises the concept of *virtual connection* in providing different treatment to each connection. The small fixed-size datagram, called *cell*, is proposed to reduce the effect of bursty traffic. Maybe, the most important concept of ATM is the differentiation of the service categories. Service categories are used by the end system to identify the type of service required and tell an ATM network to handle each traffic flow accordingly. Service categories of the ATM include *Constant Bit Rate* (CBR), *Real-Time Variable Bit Rate* (rt-VBR), *Non-Real-Time Variable Bit Rate* (nrt-VBR), *Available Bit Rate* (ABR), *Unspecified Bit Rate* (UBR), and *Guaranteed Frame Rate* (GFR) [ATM99]. Similar service categories are also redefined in other QoS-enabling frameworks and protocols as we will see later.

1.2 The existing QoS frameworks for the Internet

Despite the fact that ATM provides rich support for QoS, reliance on ATM means either constructing a second networking infrastructure for multimedia traffic or replacing the existing IP-based configuration with ATM, both of which are costly alternatives [STAL02]. Thus, developing the frameworks for supporting QoS on TCP/IP architecture is more impulsive. It is easy to see that traditional one-class, best-effort of the Internet is no longer considered efficient to support such diverse requirements. For several years, many efforts have been attempted to provide the suitable service for each type of traffic, while maximizing the utilization of the network resources for IP-based network; however, only the limited successes were achieved. Recently, Internet Engineering Task Force (IETF) has proposed two traffic management frameworks: *Integrated Services* (IntServ) and *Differentiated Services* (DiffServ).

Integrated Services (IntServ) [BRAD94] combines the concept of service categories and the Resource Reservation Protocol (RSVP). It introduces the concept of *connection* to the IP-based network and allocates the resources for each connection accordingly. Service categories in IntServ are divided into *Guaranteed Services*, *Controlled load Service*, and *Best Effort*. IntServ is a connection-oriented framework, i.e. applications that support IntServ must establish the connections in the initial phase. RSVP signaling protocol is used for reserving the resources along the path when the connection is established. The intermediate nodes must provide some form of admission control and support RSVP signaling. Each node along the path must maintain the *soft* per-flow state of each connection. Hence, the complexities and overheads are introduced to the intermediate nodes. This is not scalable since the number of per-flow states maintained by each node would have an exponential growth, as the number of nodes in the network increases. Furthermore, the applications requesting for services must support RSVP signaling to establish the connection. This is called application dependency.

While the complexities and signaling overhead due to the maintenance of state information at router make IntServ impractical to deploy, the seemingly more successful QoS framework for IP-based network is Differentiated Services (DiffServ) [NICH98], [BLAK98], [DAVI02], and [HEIN99]. DiffServ architecture defines two types of nodes: *DS boundary nodes* and *DS interior nodes*. The DS boundary nodes are responsible for classifying the traffic into different classes by marking the *Differentiated Service Code Point (DSCP)* of each packet, and performing some

traffic conditioning functions, if necessary. The service categories in DiffServ are called *Per-hop behaviors (PHBs)*. Each PHB is associated with one or more DSCP. So far, three PHBs have been standardized; *Expedited Forwarding* (EF), *Assured Forwarding* (AF), and *Best Effort* (BE). The DS interior nodes simply forward each packet according to its PHB. The DSCP actually indicates the priority of each packet. The service discrimination is done by examining the DSCP of each packet. Therefore, DiffServ can be viewed as a priority based QoS framework.

The connectionless characteristic makes DiffServ more scalable than IntServ because there is no per-flow state to be maintained and network is no longer required to support any form of resource reservations or admission controls. Furthermore, no signaling is required to be initiated from applications, which makes the framework application independent. Besides, the complexities of traffic classification and conditioning are pushed to the network edge. In next section, we will consider the case where decoupling of routing and QoS support may lead to inefficient resource management.

1.3 Motivations and goals

In DiffServ framework, the packet prioritization and PHB service discrimination only take into account *after* the routing decision has been made. While the ordinary routing decision is made based on minimizing a single metric, QoS requirements of the traffic may consist of multiple constraints, which may or may not include the constraint used by routing algorithm. The route assigned to the traffic may not be best suit its QoS requirements; therefore, there may exist alternate routes that have the resources conformed to such QoS requirements, but are not indicated by the single-metric routing algorithm. To make this claim solid, consider the following example. An OSPF routing protocol may be configured to minimize the delay between any source-destination pairs in a certain network. Suppose an AF PHB packet, which is described by some delay bound and low drop precedence, i.e. low loss rate, is finding its way through the network. The default minimum-delay route is then assigned to this packet, where it would be aggregated with other types of traffics that share the same destination. The low drop

precedence requirements would take into account *after* it is queued at the output interface of the routers. In this case, the low loss rate routes with acceptable delay bound may exist but do not indicated by the routing protocol. In addition, the application may suffer from service degradation if the minimum-delay route is a lossy path due to the error in transmission media rather than queuing drop, where low drop precedence has nothing to do with.

To address this shortcoming, routing and QoS support must be coupled to achieve better resource usage. The routing algorithm must take into account the traffic requirements of various flows and resources available along the paths in the network. Such routing algorithm is referred to as a *multi-constrained path algorithm* or *QoS routing algorithm*, as to distinguish from the traditional single-metric, one-class best-effort routing algorithm. The QoS routing problem can be formally defined as the problem that consists in finding an optimal-cost path or set of feasible cost paths from source to destination(s) subject to one or more constraints on the path. We propose the use of QoS routing for aiding DiffServ framework in suggesting the appropriate path for a better network resource utilization.

We might expect that the overall functionality of QoS routing is more complex than traditional best-effort routing. The following is the list of the desired characteristics of the QoS routing framework:

- **Scalability**: QoS routing framework must exhibit low signal and processing overheads and must be stateless, eliminating the need of maintenance of per-flow information. Connectionless mode of operation must be assumed; no resource reservation or the concept of establishing connection is allowed. Consequently, the routing must be application independent and transparent to the higher layer; no direct signaling from the application is needed to trigger the routing process.
- **Efficiency**: the framework should provide better resources management through coupling of routing decision and QoS support. As we mention earlier, using service differentiation based on appropriate scheduling policies may lead to inefficient resource management since different traffic may be routed along the same best-effort path. QoS routing framework must exercise the efficient usage of resources while taking into account the multiple QoS requirements of traffic for route selection.

- **Robustness**: other than the loop-freeness property of the routing protocol, dynamic variation of the load should not cause path oscillation.

The above requirements serve as a guideline for developing our QoS routing framework. The traditional routing mechanisms, such OSPF and BGP, employ only single cost metric and thus routing based on multiple constraints is not achievable. Either enhanced version of an existing routing protocols or a new one is required. There has been many proposed works in the literature for this issue (see chapter 2.0). For example, the QoS routing mechanism extension to OSPF [APOS99b] offers the route selection based on two metrics; namely, available bandwidth and hop-count. In essence, the route selection is done in such a way that minimum hop-count path is selected from a set of feasible available bandwidth paths. The rule for selecting among these paths is meant to balance load as well as maximize the likelihood that the required bandwidth is indeed available. In general, there is a trade-off between the protocol overhead of frequent updates and the accuracy of the path selection results. For large scale network, the frequent updates make the protocol neither practical nor feasible to deliver the routing decision. The scalable generalized QoS routing framework is yet to be discovered. This leads to our fundamental research question:

> ***"Is it possible to develop a scalable and robust routing architecture to support QoS requirements of the current emerging multimedia applications in Internet environments?"***

We propose the treatment to the above question in the next section.

1.4 The proposed methodology

Scalability and efficiency of the QoS routing architecture may be achieve by introducing the concept of *network clustering* and *selective probing scheme*. Network clustering is done in order

to reduce the number of nodes that will participate in the routing process, specifically, the metrics acquisition process. Nodes that share the same *class of equivalence* are grouped together and form a *cluster.* Class of equivalence can be defined in terms of any metrics. In this thesis we focus only on the *delay* metric such that the nodes are said to share the same class of equivalence if, and only if, the delay variations among them are bounded to some value, usually specified by parameter d_c. A cluster itself can be viewed as a logical node, called *meta-node*. The network topology of the meta-nodes is called a *meta-graph*, which, in turn, is derived from the physical connectivity of the clusters. The concept of meta-nodes and meta-graph is shown in Figure 1-1.

The selective probing scheme is introduced to reduce the routing signaling overhead. Once the network is clustered and the meta-graph is derived, the metrics acquisition process can be done among meta-nodes, in per-cluster basis, in contrast to per-node basis. For each cluster, or meta-node, we locate a *representative node* that will represent other nodes, in its own cluster, in terms of QoS metrics. The metrics acquisition is done by sending the *probe* signal to the representative node of the cluster of interest to acquire its QoS metrics information. Then, the QoS metrics of the rest of the nodes in the cluster can be *inferred* from this probed information. Our assumption is that: there is a correlation between QoS metrics. Specifically, we assume that the nodes that share delay-based class of equivalence also have some similarities in the measures of other metrics, e.g. jitter, loss rate, available bandwidth.

Once the metrics information has been acquired from the meta-nodes and metric estimation has been done for the rest of the nodes in the network, the path selection algorithm can be run to attain the appropriate paths that satisfy the QoS requirements of the underlying applications. Figure 1-2 summarizes the overall process of network clustering and selective probing scheme. Note that the cycle of metrics acquisition, metric estimation, and path selection can be done periodically or upon request. Network will also need to be re-clustered to update the meta-graph topology, however, in a larger unit of time. The concepts of clustering and selective probing will be revealed in more details in chapter 4.0.

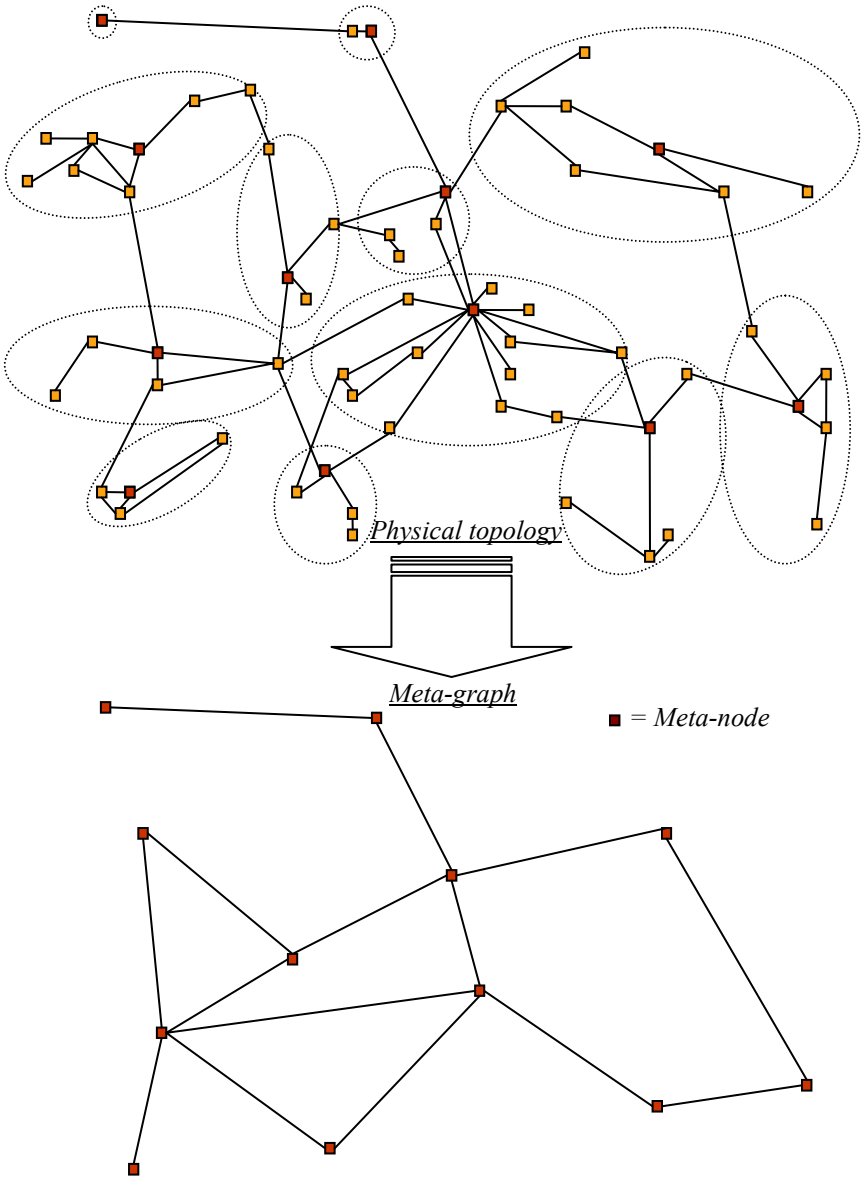

Figure 1-1: The concept of *network clustering* and *meta-graph*

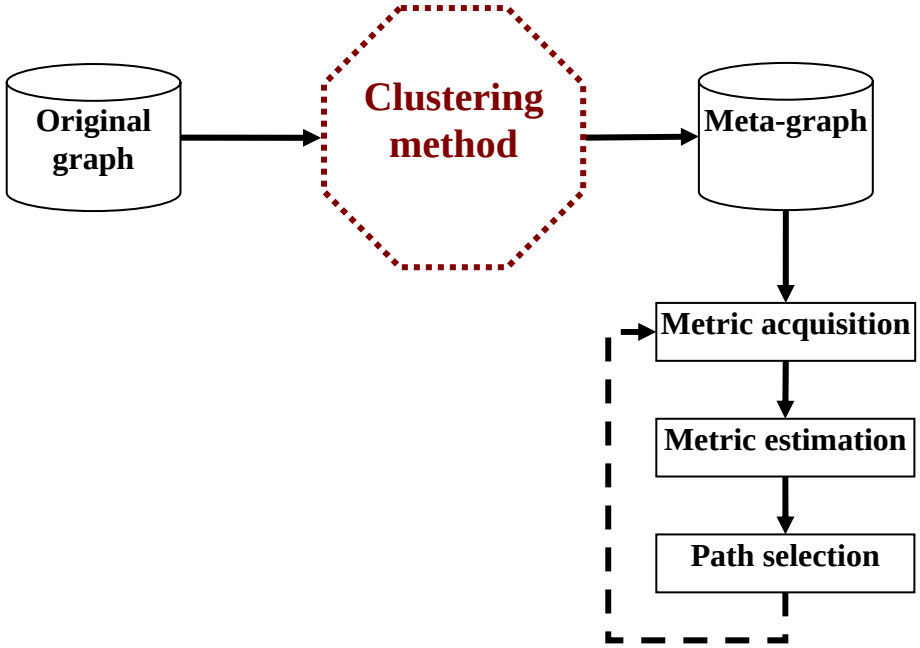

Figure 1-2: The clustering and selective probing framework

Once the network of n nodes is clustered into k clusters, the routing signaling overhead is reduced by the factor of n / k. For scalability, we need the number of clusters k to be small. However, having a smaller number of clusters means the clusters must yield the larger average size, which results in higher error in metrics estimation. Remember that the accuracy of this metric acquisition scheme depends on how close nodes are correlated in terms of delay. Our study focuses on the methods that efficiently cluster the network in order to reduce the routing signaling overhead while minimize the metrics estimation error. We propose the new clustering approach and evaluate its performance in chapter 3.0. The results show that our approach outperforms the currently used approach, at least based on our set of performance parameters.

Since we focus on the QoS routing for the wide area network, we exploit the power-law and small-world properties of the Internet in simulating various instances of graphs that represents the Internet. Detail of the power laws of the Internet and the small-world phenomena will be discussed in chapter 2.0.

1.5 Major contributions of the thesis

The thesis has the following contributions:

1. **Selective probing architecture**: we propose the concept of metrics probing for scalable QoS routing framework. So far, there is no obvious method to do such selective probing scheme. A few works in the literature claimed the term selective probing. None of them has shown intensive studies on the problem. For example, methods of selective probing introduced as a part of works in [CHEN98a], [LEE00] and [LEE01] are, basically, flooding the routing packets selectively to explore the feasible paths. Their meaning of the term selective probing is different than ours in the sense that their *selective* is done on paths, while ours is done on nodes. Furthermore, to the best of our knowledge, no work has been considered the problem of clustering the network for the purpose of scalable routing information retrieval. Therefore, this work will concentrate on the problem of optimally locating clusters and their representative nodes for the selective probing QoS routing architecture.
2. **Clustering methods and core routers selection**: we propose and evaluate the method of delay-based clustering and selecting the representative node, a core router, for each cluster. The delay-based clustering scheme can appropriately be used with various tasks other than routing, such as network infrastructures placement problems, content distribution network (CDN) resources management, and network distance estimation for overlay network.
3. **QoS metrics estimation**: we provide a simple method of QoS metrics estimation for large scale network, which aims to reduce the signaling and processing overheads of the routing protocol.

In this research, we evaluate mathematical models currently used for clustering and propose a new model that is more suitable for our selective probing architecture. Then we implement and evaluate our new model along with the existing models. Results indicate that the proposed model outperforms the existing models. The experimental results and observations make us give useful suggestions on parameters settings for the algorithm.

1.6 Thesis organization

This thesis is organized as follows. The background related to our work is given in chapter 2.0. We will discuss three areas, namely; QoS routing, power laws of the Internet and discrete location models. A brief review on the related works is also given at the end of the chapter. Chapter 3.0 focuses on the problem of clustering the network, including some experimental results and analyses. Chapter 4.0 describes the framework for selective probing scheme. We conclude our work in chapter 5.0.

2.0 BACKGROUND AND RELATED WORKS

In this chapter, we review some fundamental backgrounds required for our selective probing scheme. We start with an overview on QoS routing and introduction to the power laws of the Internet and the relating issues on the Internet topology generators. Then, we consider four well-known discrete location models that are used for clustering, which is the focused area of this research. Finally, we review the works in the literature relating to our work.

2.1 QoS Routing

To meet the QoS requirements of multimedia applications, the stateless, priority-based DiffServ framework has been proposed. However, as it has been discussed in the previous chapter, using DiffServ solely, without any collaboration with routing, may lead to inefficient resource utilization. Moreover, traffic may be routed through a path which cannot satisfy the application's QoS requirements, even when such a path exists in the network. To address this shortcoming, we propose an efficient QoS-aware routing framework in support of the DiffServ framework to achieve better resource utilization.

In general, routing (QoS-based or not) involves two entities: routing protocols and routing algorithms. Routing protocols capture the network state information (e.g. available resources) and disseminate it throughout the network, while routing algorithms use this information to compute appropriate paths. (For example, Dijkstra's is a routing algorithm, while OSPF is a routing protocol.) While current best-effort routing simply performs these tasks based on a single, relatively static measure, QoS routing takes into account both the traffic's requirements and the availability of network resources. As a result, QoS routing has to deal with the

challenging issues that are not presented in the best-effort routing, including scalable dissemination of dynamic (state-dependent) information, state aggregation, and computation of constrained paths [KUIP02].

QoS routing must extend the current routing paradigm in several ways. It must support traffic using different classes of services, or some other means of service differentiation, and multiple paths between nodes pairs will have to be calculated [CRAW98]. For the Internet, the development of efficient QoS routing algorithms and protocols is still an open issue that needs to be investigated further. Recently, there have been many proposals for QoS-based frameworks (e.g. IntServ, DiffServ, constraint-based MPLS), QoS routing algorithms (mostly heuristics, see [KUIP02]) and QoS routing protocols (e.g. Q-OSPF [GUER97], [APOS99b], PNNI [ATM02]). In this section, we introduce the concepts of QoS routing algorithm and QoS routing protocol and discuss works related to these concepts.

2.1.1 QoS routing algorithms

Before we give the definition of the QoS routing, we first point out that the QoS measures can either be *additive* (e.g. delay and jitter), *multiplicative* (e.g. packet loss rate), or *min/max* (e.g. available bandwidth and policy flags) [WANG95]. For additive measures, the path weight of such measure is the sum of QoS measure over links along the path. The multiplicative measures can be transformed into additive measures by using logarithm. The min/max measures can easily be treated by *topology filtering*, which is done by omitting all the paths that do not satisfy the QoS requirements. Without loss of generality, we assume all QoS measures to be additive.

In general, the network topology can be mathematically represented by a graph, denoted by a function *G(N, E)*, where routers are represented by nodes (*N*) and links are represented by the edges (*E*) of graph. The problem of QoS routing in Wide Area Networks can be formally defined as a *Multi-Constrained Path (MCP) problem*, as follows [KUIP03]:

> ***Definition 2.1:*** *Multi-Constrained Path (MCP) problem. Consider a network G(N, E). Each link* $(u, v) \in E$ *is specified by a link weight vector with components m additive QoS weights* $w_i(u, v) \geq 0$, $i = 1,\ldots,m$. *Given m constants* L_i, $i = 1,\ldots, m$, *the problem is to find a path P from a source node* ***s*** *to a destination node* ***d*** *such that*

$$w_i(P) = \sum_{(u,v)\in P} w_i(u,v) \le L_i \qquad \text{for } i = 1,\ldots,m.$$

A path that satisfies the above condition is said to be *feasible.* Note that there may be multiple feasible paths between ***s*** and ***d***. A modified (and more difficult) version of the MCP problem is to retrieve the shortest *"length"* path among the set of feasible paths. This problem is known as the *multi-constrained optimal path* (MCOP) problem, and is attained by adding second condition on the path *P* in Definition 2.1:

$$l(P) \le l(Q)$$

for any feasible path *Q* between source and destination, where *l(·)* is a path length (or cost) function. A solution to the MCOP problem is also a solution to the MCP problem, but not necessarily vice versa. Considerable works in the literature have focused on a special case of the MCOP problem known as the *restrict shortest path* (*RSP*) problem, where the goal is to find the least cost path among those that satisfy only one constraint denoted by *d*, which bounds the permissible delay of a path. For further details, see [KUIP02].

The problem of QoS routing has been investigated extensively. The work described in [JAFF84] is among the first to propose an algorithm for finding paths that satisfy multiple constraints. The MCP problem and its variations are known to be NP-complete [GARE79] was the first to list the MCP as being NP-complete but did not provide a proof. [WANG96] provided the proof for the case where the number of constraints $m \ge 2$. This suggests that the MCP problem is intractable for large-scale networks. Therefore, many heuristics and approximation algorithms have been proposed for this problem. The *routing algorithms* referred in the following context are essentially the approximation algorithms or heuristics.

[MA97] provided a comparison of four routing algorithms, which are widest-shortest path proposed by [GUER97], shortest-widest path proposed by [WANG96], shortest-distance path proposed by [MA96] and dynamic-alternative path. A comprehensive survey on QoS routing algorithms was given by [CHEN98b]. But they focused on network models in virtual circuit mode. A more recent overview on constraint-based path selection algorithms was given by

[KUIP02]. This work discussed a collection of QoS routing algorithms for both MCP problems and RSP/MCOP problems. [SOBR02] reviewed the algebra used in QoS path computation. [KORK03b] and [SIAC03] proposed general approximation algorithms for QoS routing.

The inaccuracy of the routing information has been raised into consideration since it may lead to faulty path selection. In the context of QoS routing, traffic may be routed to the path with inadequate available resources, even if the path that satisfies the traffic's QoS requirements exists. Due to the presence of uncertain network state information, inaccuracy of the routing information is likely to be occurred. The impact of the inaccurate routing information on the QoS routing were studied in [GUER99], [KORK03a], and [CHEN98c]. The inaccuracy of the routing information may be mitigated by frequent updates, which increase the routing overheads and complexities. Therefore, the accuracy of the routing information is a tradeoff to scalability.

The scalable routing information retrieval may be accomplished by employing hierarchical routing scheme. The well-known hierarchical routing architecture is the *Private Network-Network Interface*, (PNNI) [ATM02], used in ATM network. In context of conventional datagram operation routing, the works by [LUI00] and [LEE03] have addressed the hierarchical routing with QoS supports.

Nevertheless, most of the works were done in the context of a connection-oriented model, which requires the network to maintain states information of each connection. Very small portion of works in the literature assume connectionless model, for example, the work proposed by [WANG02], which discussed about a differentiated hop-by-hop routing algorithm based on the DiffServ framework. In this work, the connectionless model must be assumed since we are working within the Internet framework with a goal to be compatible with DiffServ QoS framework for scalability.

2.1.2 QoS routing protocols

In contrast to the extensive study in QoS routing algorithms, the research on QoS routing protocols has a comparably very small number of works. Some QoS routing protocols are coupled with specific QoS routing algorithms (such as the proprietary resource discovery or resource reservation protocols). However, none has addressed the mechanism of the protocol in details. Most of the QoS routing protocols were proposed in a form of an extension to the

existing routing protocol, such as OSPF [MOY94], e.g., [APOS99a], [APOS99b], and [GUER97], and BGP [REKH95], e.g., [XIAO02]. The extensions to the existing protocol have many benefits. First, it guarantees the compatibility with the existing protocol. Second, the new protocol can be quickly adapted in practice. Lastly, it is easier to design since it does not require redesigning the entire protocol.

Designing a QoS routing protocol raises two major concerns which are difficult to achieve simultaneously, accuracy and scalability. Issues related to the accuracy of the routing information have been addressed in the previous section. The scalability is brought into consideration since the network is dynamic and multiple metrics are collected from the network. Measurements for some of these metrics may require further overhead. For example, metrics such as delay or delay jitter may require periodic probing and appropriate averaging techniques over time for a single measure, which introduces some packet and time overheads in network. At this point, we must address again that scalability is a very important issue for a large-scale network.

Our work is an attempt to deal with the tradeoff between scalability and accuracy. We propose the QoS routing protocol in an alternative way. For scalability, we cluster the network of n nodes into k clusters, where $k << n$, and probe for the QoS measures from the center of each cluster. The QoS measures of the rest of the nodes in the network can then be estimated from the knowledge of the QoS measures of these k center nodes. The number of QoS routing queries is then reduced from a factor of n to only a factor of k. The scalability of the scheme is then determined by the ratio of k and n. The accuracy of the QoS measures estimation can be achieved by minimizing the diameter of clusters, which may be represented by delay. However, since the two quantities are tradeoffs, we must find the clustering method that yields the optimal point between accuracy and scalability.

Optimal clustering the Internet, which topology is unknown and dynamic, is a hard task. However, recent researches have indicated that the topology of the Internet is not totally randomized. Aside from its constantly changes in topology, some properties of the Internet hold from time to time. These properties are known as power laws of the Internet, which we will discuss in the next section.

2.2 Power laws of the Internet

2.2.1 The power laws

Power laws have been first discovered by a Harvard linguistic professor, George K. Zipf, who introduced the Zipf law [ZIPF32], which states that the frequency of use of the *n*th-most-frequently-used word in any natural language is approximately inversely proportional to *n*. Essentially, the law above can be written in a form of exponential equation, with a constant exponent value, depending on the given natural language. This constant exponent captures the static property of the given language, no matter how the language changes. This property is later commonly observed in many kinds of dynamic natural phenomena, including the Internet topology.

In 1999, [FALO99] introduced three power laws for the topology of the Internet. Power laws are the expressions of the form $y \propto x^a$, where an exponential a is a constant, x and y are the measures of interest, and $\propto$ stands for "proportional to." The observations showed that some of those exponents do not change significantly over time. The importance of the discovery of the existence of power laws is the fact that there is *some* exponent for each graph instance. Specifically, the graph of the Internet topology has its own unique set of such exponents. Power laws can be viewed as a set of quantitative statistical models that precisely capture the highly skewed, heavy-tailed, distributions of the topological graph properties of the Internet. The following is a summary of three power laws for AS-level Internet topology:

Power-law 1 (Rank Exponent): Given a graph, the degree d_v of a node v is proportional to the rank of the node r_v to the power of a constant $\mathcal{R}$

$$d_v \propto r_v^{\mathcal{R}}$$

Where rank of the node r_v is obtained by sorting nodes in decreasing order of degree d_v.

Power-law 2 (Degree Exponent): Given a graph, the CCDF D_d of a degree d is proportional to the degree to the power of a constant D

$$D_d \propto d^D$$

Power-law 3 (Eigen Exponent): Given a graph, the eigenvalues λ_i are proportional to the order i to the power of a constant ε

$$\lambda_i \propto i^\varepsilon$$

Where the eigenvalues of a graph are the eigenvalues of its adjacency matrix and i is the order of λ_i in the decreasing sequence of eigenvalues.

Power laws above were the revised studies by [SIGA03], as for the AS-level Internet topology. There are many interesting results from the studies of the power laws. We will discuss some of them. The rank exponent for the Internet has been observed to be around -0.81, the degree exponent is around -1.12 and the eigenvalue exponent is around -0.47. There are also some observations on the relationships of these exponents. Theoretically, the relationship between rank exponent and degree exponent is equal to $\mathcal{R} = 1/D$. The eigenvalue exponent is approximately half of the degree exponent, i.e. $D = 2\varepsilon$.

Furthermore, [FALO99] have developed useful formulas for estimating the effective diameter, the average neighborhood size, and the number of edges of the Internet. The effective diameter, intuitively, is the number of hops that two nodes are located within, with high probability. The average neighborhood size is the average number of nodes within some given hopcounts. The estimated number of edges is calculated from number of nodes in the network and rank exponent, $\mathcal{R}$.

The second power law indicates that the nodes with lower degree outnumber the nodes with higher degree. This implies that the lower degree nodes, especially the 1-degree nodes, are attached to the higher degree nodes with many-to-one relationship. The connectivity of the Internet topology is then not totally randomized, but exhibits some form of cluster. This assumption has been supported by the work of [BU02], which observed that the Internet topology also exhibits the *small-world* phenomena. The small-world phenomena, also known as

six degrees of separation, has been shown to exhibit in many self-organizing networking systems, including the power grid of the western United States, the collaboration graph of film actors, and the Internet topology. Essentially, the connection topology of a small-world network is neither completely regular nor completely random, but it lies somewhere between these two extremes. The work by [WATT98] addresses two important structural properties that capture the characteristic of these small-world graphs, which are *characteristic path length* and *clustering coefficient*.

[TAUR01] studied the power laws and topological connectivity, and developed a simple graphical conceptual model for the better understanding on the Internet topology, called *jellyfish* model. This conceptual model exhibits the loose hierarchy of the Internet.

2.2.2 The power-law based topology generators

The discovery of the power laws of the Internet brought a revision of the graph generation models in the networking community. Realizing that the Internet topology has some specific properties addressed by power laws, [FALO99] suggested that the power laws can be used, as one of the criteria, for validating the generated Internet graphs.

There are two kinds of graph generation tools: Structural and degree-based generators. The structural generators are based on the idea that the Internet is hierarchical. The example of these generators is the famous GT-ITM (Georgia Tech Internetwork Topology Models) [CALV97], [ZEGU96], which is now integrated in the *NS-2* package [NS2]. This type of generators has been disproved by the time of the discovery of the power laws because the degree distribution produced by the structural generators is not power-law. The degree-based generator is more promising to deliver the power-law graph. The degree-based generators, such as [BARA99], [MEDI00], [MEDI01], [WINI03], [BU02] and [MAGO02], among others, try to match the Internet's degree distribution, without any concern on the Internet's hierarchical structure. Nowadays, it's widely accepted that the degree-based generators are significantly superior to structural generators for the large-scale network, say larger than 1,000 nodes. The hierarchy presented in the measured networks is looser and less strict than those generated from the structural generators, and this is well captured by the hierarchical structure in degree-based

generators. This may explain why these generators better match the measured topologies. The detailed discussion on the two types of network topology generators can be found in [TANG02].

[BU02] are among the first who adopted the use of the *characteristic path length* and *clustering coefficient* of small-world phenomena [WATT98] into the Internet topology generator. They observed that the previous algorithms have difficulty in generating topologies that have these values matched to the real Internet topology. The work concluded by proposing an improved algorithm that does a better job matching the two quantities than previous generators.

As we have discussed so far, the validation of the Internet graph takes into account only the topological connectivity issue. Today's Internet topology generators pay no attention to the weights on the graph, which may be used to represent either delay or bandwidth of links on the Internet. This is because the weight information on the Internet is dynamic and is difficult to obtain. However, our work requires the weight to be presented on the generated graphs. In the following section, we will address this difficulty and propose the sensitivity analysis as our study method.

2.2.3 Difficulties in using the power-law graphs

The study of power laws yields the statistical models that can be used to decide the conformance of the generated graph to the real Internet topology. Power laws can be viewed as the properties, among others, of a class of graphs. However, power laws focus only on the number of nodes and the connectivity properties of the graph, without any consideration on the weights of the edges of the graph. Consequently, most of the graphs generated by the power-law based network topology generators can provide neither the information about delay nor bandwidth on the links. The delay and bandwidth distributions of the real Internet are needed to be investigated further, in order to emulate the Internet graph more completely. This is a hard issue since even the captured routing information, i.e. from BGP, usually does not provide this information.

The measured Internet topology is provided by [CAIDA]. Recent measurement reveals that the Internet consists of approximately 200,000 Autonomous Systems (ASs) and power laws still hold. However, only the connectivity information is captured, no information on the weights (delay, bandwidth, etc.) on the links is supplied. Furthermore, the exceedingly large real Internet topology is impractical to use in the early stage of our study. The techniques for summarizing the

input data must be further developed before we can apply our selective probing scheme. Therefore, the smaller synthesized power-law Internet topology is preferred.

In our work, we use INET 3.0, an AS-level internet topology generator that obeys both power laws and small-world phenomena. The conceptual idea and analysis on the validation of the INET 3.0 are described in [WINI03]. Like other internet topology generator, INET 3.0 only provides the connectivity information; the generated topologies do not have any information pertaining to the weights of the edges of graph. Besides, as for today, we do not have a measured Internet topology map with links labeled by bandwidth or delay. That is, we do not know the exact delay distribution of the Internet. Rather, we do the sensitivity analysis on the delay distribution. For each graph generated from INET 3.0, we associate it with one of the four well-known distributions, which are uniformed, normal, exponential and power-law (or heavy-tailed) delay distributions.

2.3 Discrete location models

As described earlier, the efficiency of the selective probing scheme depends on how the network is clustered. Given a graph $G(n, e)$, we need to partition it into k clusters, where $k < n$. Each cluster is a delay-based class of equivalence. That is, let i be the center of cluster C, node j is a member of cluster C if and only if $d(i, j) \leq d_c$, where $d(\cdot)$ is a delay function denotes the delay between node i to node j, sometimes written as d_{ij}, and d_c is the maximum allowable delay bound. In our selective probing scheme, node i that represents the center of a cluster is called ***anchor*** (and is referred as *facility* in discrete location models). d_c is a coverage distance of a cluster. For delay-based clustering, d_c is a pre-defined maximum delay bound of a cluster and is referred to as a ***radius*** of a cluster, provided that anchor is a center of a cluster. (We avoid using the term *cluster size* since it may lead to the meaning of number of nodes in cluster.)

By partitioning the network into clusters, the distance of any two nodes in the network can be estimated by the distance between their respective anchors, assuming the distances within clusters are negligible. Basically, the *radius* d_c of each cluster should be as small as possible to

minimize the error from the estimation. On the other hand, the small number of clusters k is also desired for scalability since it reflects the number of routing entries that need to be maintained at each node.

However, the *radius* d_c has an inverse relationship to the number of clusters k. That is, if the radius d_c of clusters is reduced, more number of clusters k must be located, and vice versa. Therefore, we need to balance the clusters' radiuses and number of clusters, in order to minimize the estimation error, while maintaining scalability. Since the two criteria cannot be minimized simultaneously, we need the optimal solution of one criterion given that another criterion is a constraint. The term *'optimal'* is defined in a mathematical sense so that the quantifiable objectives must be defined and algorithms for solving them must be identified.

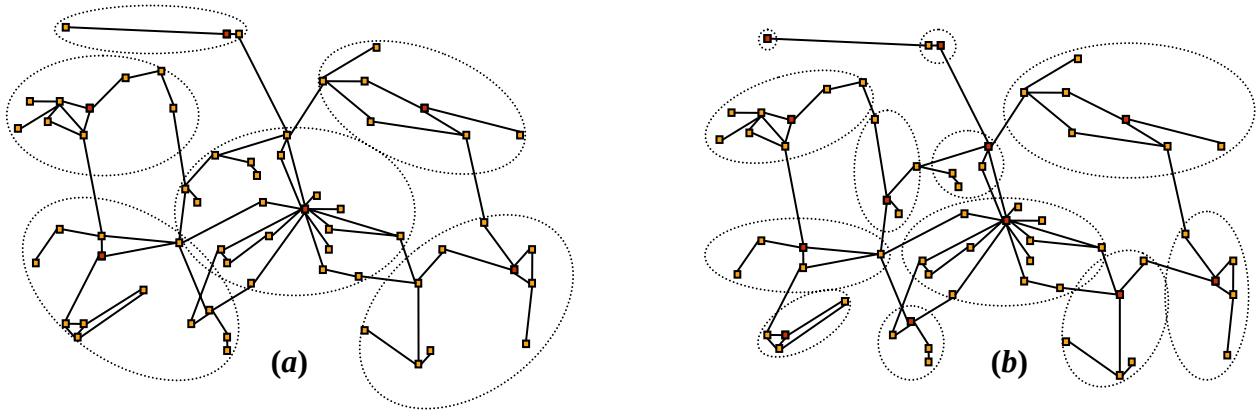

Figure 2-1: The clustered networks with (*a*) *k* = 6, (*b*) *k* = 12.

Finding the suitable mathematical models is then one of the key questions to our research. Before proceeding to the mathematical models, we would like to briefly describe the characteristics of the problem of clustering involving in the selective probing scheme, as follows. (1) The clustering should be done so that the nodes within the same cluster share the same delay-based class of equivalence. (2) Distance between a pair of anchors determines the estimated delay of any pair of nodes located in the respective clusters. (3) The average *radius* of the clusters and the number of clusters have an inverse relationship. (4) Given a mathematical model for the clustering problem, the solving algorithms must take either number of cluster k or radius

d_c as an input, and return the other as an output. In addition, the algorithms must determine the optimal location of the clusters. Figure 2-1 illustrates the basic idea of clustering a graph using two different value of k. In Figure 2-1(*a*), k is 6 and Figure 2-1(*b*), k is 12. We can see that the average radius of Figure 2-1(*a*) is larger than Figure 2-1(*b*).

2.3.1 Four discrete location models

The discrete location theory has long been a key subject of study in *Operation Research*. Location theory involves with the location decisions using mathematically formulated location models. The common scenario is to locate the facilities, i.e. bank branches, retail stores, fire stations, or ambulances, onto the network of roads, which is represented by graph. In this section, we introduce four discrete location models for finding desirable or optimal facility locations. Each model differently defines quantifiable objective function that reflects the term *'optimal'* in various means. Generally, the location models have been developed to address a number of questions including [DASK95]:

a. How many facilities should be sited?
b. Where should each facility be located?
c. How large should each facility be?
d. How should demand for the facilities' services be allocated to the facilities?

The fourth question comes into concern when we consider the capacitated version of the models, where each facility has explicit capacity limit. However, we will be interested in only the uncapacitated version, hence; this question can be ignored.

All the discrete location models that we will discuss later can be formulated as integer linear programming problems. As such, one naive way to solve these problems is to relax the integer requirement on the decision variables, solve the resulting integer linear programming problem and then employ a branch-and-bound strategy to force integrality of the variables that must be integer. Unfortunately, each problem has been proved NP-hard. That is, the problems cannot be solved deterministically within polynomial time of the input length, unless $P = NP$. Solving the

integer linear programming problems directly is infeasible. The more efficient approach is using the approximation algorithms or heuristics.

In following, we review the four discrete location models: set covering, *k*-center, *k*-median, and facility location models. For each problem model, the problem is formulated in the standard form of integer linear programming. Then we review the approximation algorithms proposed in the literature. The detail of an approximation algorithms are given at the end of each section.

2.3.1.1 The Set Covering Problem

The set covering is perhaps the simplest model known and used in discrete location models. The objective of the set covering problem is to find a minimum cost set of facilities from among a finite set of candidate facilities so that every demand node is covered by at least one facility. This can be mathematically formulated in the typical integer linear programming problem as follows:

$$\textbf{MINIMIZE} \qquad \sum_{i \in F} x_i$$

$$\textbf{SUBJECT TO:} \quad \forall j: \quad \sum_{i \in F} a_{ij} x_i \geq 1$$

$$\forall i: \quad x_i \in \{0,1\}$$

$$\forall i, j: \quad a_{ij} \in \{0,1\}$$

x_i is a decision variable and has value (set to 1) if and only if a facility i in F, where F is a set of facilities, is located (or selected). The objective function minimizes the total cost of the facilities that are selected. This is a special case, where all of the facility costs are identical, sometimes called *unweighted* or *unicost* set covering problem. Hence, the problem is reduced to minimizing the number of selected facilities. Also note that, this is an uncapacitated version since all facilities are assumed having unlimited capacity.

a_{ij} is a connectivity matrix and has value (set to 1) if and only if facility i covers node j. Distance between nodes i and j denotes by d_{ij}. Given a coverage distance d_c, facility i is said to cover node j ($a_{ij} = 1$) if and only if $d_{ij} \leq d_c$, otherwise $a_{ij} = 0$. The distance d_{ij} can be calculated from the input graph using one of the distance metric functions, e.g. shortest path or Euclidean

distance metrics. The coverage distance d_c is exogenously defined. Therefore, the first constraint ensures each node j must be covered by at least one facility. The second and third constraints indicate that x_i and a_{ij} are Boolean.

Solving the integer linear programming directly is infeasible because the set covering problem on a general graph is NP-complete [GARE79]. [DASK95] outlined three techniques for reducing the size of the problem, i.e. reducing the size of connectivity matrix a_{ij}, which are *column reduction rule*, *row reduction rule*, and *second row reduction rule*. However, these rules cannot guarantee the amount of reduction. In some cases, the problem size cannot be reduced. After applied the rules, the problem is then solved by relaxing the integrality constraints to the nonnegativity constraints. To ensure the all-integer solution, an additional technique will generally be required. The common approach is to use branch-and-bound. Further details in techniques for solving integer linear programming are briefly described in [DASK95].

The more efficient approaches are the approximation algorithms that run in polynomial time and deliver solutions that are close to optimal. The efficiency is a tradeoff between running time and approximation factor. The approximation factor gives a lower bound of an approximated ratio of the algorithm's solution over the optimal solution. A small constant approximation factor is desirable; the approximation factor of 1 implies that the algorithm always returns the optimal solution.

Despite the extensive studies on the set covering problem, the best approximation algorithm known is greedy-based, which yields an approximation factor of $ln\ n$. [JOHN74] was among the first who gave the greedy algorithms for the set covering problem, which yield an approximation factor of $ln\ n$. [GROS97] conducted a comparative study of nine different approximation algorithms for the set covering problem, including several greedy variants, fractional relaxations, randomized algorithms and a neural network-based algorithm. The study was done on a set of random-generated problem sets, where the optimal solutions were known [ORLIB]. The greedy-based algorithms (both randomized and deterministic variants) yielded the best results. A constant approximation factor algorithm has not yet been found in the literature. [LUND94] showed hardness of approximation within a ratio of $(log_2 n) / 2 \approx 0.72\ ln\ n$. [FEIG98] proved that the best achievable approximation factor is $(1 - o(1))\ ln\ n$. [ALON03] recently studied the online set covering problem, where the input elements are given one-by-one.

The greedy algorithm for set covering problem outlined by [JOHN74], requiring running time proportional to n^2 and approximation factor of $ln\ n$, where n is an input length, is as follows:

1. $SUB := \emptyset$; $LEFT = F$; $UNCOV = \bigcup_{S \in F} S$;
2. while $UNCOV \neq \emptyset$ do
   Choose $S' \in LEFT$, which minimizes the ratio $|S'\text{-}UNCOV'| \,/\, |S' \cap UNCOV|$;
   $SUB := SUB \cup \{S'\}$;
   $UNCOV := UNCOV - S'$;
   $LEFT := LEFT - \{S'\}$;
   end while
3. return SUB.

The algorithm is presented using the set notations. F is set of clusters, which can be overlapped. $S \in F$ is a cluster consisting of nodes. At each iteration, the algorithm adds a cluster S' that least overlaps the existing solution and returns when every node is covered.

2.3.1.2 The *k*-center Problem

In the set covering model, a proper coverage distance d_c must be exogenously pre-specified before hand. Solving the set covering problem gives us the locations and the number of clusters that are required to cover all the nodes in the network. Alternatively, in this section, we introduce a model that minimizes the coverage distance d_c such that each node is covered within the endogenously determined distance by one of the facilities. The model is called *k-center* problem, where k is the maximum allowable number of facilities that can be located. The model is known as a minimax problem, since it minimizes the maximum distance between a node and the nearest facility.

k-center problem is further divided into two sub problems: (1) the problem in which the facility can be located anywhere on the network (i.e., on the nodes and on the links of the network) and (2) the problem in which facilities can be located only on the nodes of the network. The former is known as the *absolute center problem* and the latter is known as *vertex center problem*. Note that, only the absolute center problem can guarantee to give optimal solution. To see this, consider the network in Figure 2-2. For the *absolute 1*-center problem, locating single

facility in the middle of the link is optimal and the maximum distance from either node to the facility is only *l/2*. On the other hand, in the case of *vertex* 1-center problem, the facility must be located on either node and the maximum distance from the other node to the facility is *l*. However, since we are dealing with the data network, we will be interested in only the vertex *k*-center problem.

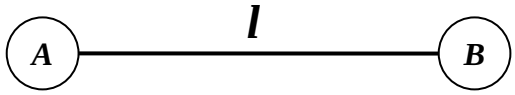

Figure 2-2: Example network for *absolute* and *vertex* 1-center problems.

The integer linear programming version of vertex *k*-center problem is formulated as follows:

$$\textbf{MINIMIZE} \qquad d_c$$

$$\textbf{SUBJECT TO:} \quad \forall j \in C: \qquad d_c \geq \sum_{i \in F} d_{ij} x_{ij}$$

$$\forall i \in F, j \in C: \quad y_i - x_{ij} \geq 0$$

$$\sum_{i \in F} y_i \leq k$$

$$\forall i \in F: \qquad y_i \in \{0,1\}$$

$$\forall i \in F, j \in C: \quad x_{ij} \in \{0,1\}$$

The objective function is the minimax problem; it minimizes d_c, the maximum distance between a node and the nearest facility. F is a set of facilities and C is a set of all nodes in network. d_{ij} denotes the distance between facility i and node j. y_i and x_{ij} are the decision variables. y_i equals to 1 if and only if a facility i is located (or selected). x_{ij} equals to 1 if and only if node j is served by the facility i. The first constraint ensures that d_c must be greater than or equal to the distance between any node j and the facility i to which it is assigned. The second constraint states that node j cannot be assigned to a facility i unless a facility i is located. The third constraint stipulates that at most k facilities can be located.

k-center problem can be viewed as a dual to the set covering problem. It takes the number of clusters as an input and tries to minimize the maximum coverage distance of the clusters, while the set covering problem takes the maximum coverage distance as an input and tries to minimize the number of clusters.

Vertex *k*-center problems on a tree can be solved deterministically within polynomial time. However, the vertex *k*-center problem on a general graph is NP-complete [GARE79]. For the approximation algorithm, [DASK95] suggests the binary search technique for solving the *k*-center problem. The procedure works as follows. Select initial lower and upper bounds on the value of the vertex *k*-center objective function. Solve the set covering problem using the average of the lower and upper bounds on the objective function as the coverage distance (rounding the average down to the largest integer less than or equal to the average). If the number of facilities needed to cover all nodes at the distance (the objective of the set covering model) is less than or equal to *k*, reset the upper bound on the value of *k*-center objective function to the coverage distance that was just used; if the number of facilities needed is greater than *k*, reset the lower bound to the coverage distance that was just used plus 1. If the lower and upper bound are equal, stop; if not, solve the set covering with a coverage distance equal to the average of the lower and upper bounds (rounded as before) and continue the process. Basically, the algorithm guesses the value of the *k*-center objective function by solving multiple set covering problems. Further details and discussions on the topic of *k*-center problem can be found in [DASK95].

2.3.1.3 The *k*-median Problem

The two discrete location models discussed so far, the covering and center problems, assume that a node receives complete benefits from a facility if it is within the coverage distance and no benefits if the distance between the node and the nearest facility exceeds the coverage distance. In many cases, however, the cost associated with a node / facility pair increases gradually with the distance between the node and the nearest facility. For example, in our delay-based selective probing scheme, the distance represents delay and the accuracy of the delay estimation depends on the delay between the node and its anchor. The cost function associated with the delay between node / anchor pair is called a *connection cost*. The connection cost increases as delay increases. Generally, cost function is a linear function of delay. For the problems that are

sensitive to the Quality of Service, the cost functions should be included into the consideration when the location decision is made.

In this section, we introduce *k*-median problems. The *k*-median problem locates *k* facilities onto the network so that the total connection cost is minimized. This problem may be formulated as follows:

$$\textbf{MINIMIZE} \qquad \sum_{i \in F,\, j \in C} c_{ij} x_{ij}$$

$$\textbf{SUBJECT TO:} \quad \forall j \in C: \qquad \sum_{i \in F} x_{ij} \geq 1$$

$$\forall i \in F, j \in C: \quad y_i - x_{ij} \geq 0$$

$$\sum_{i \in F} y_i \leq k$$

$$\forall i \in F, j \in C: \quad x_{ij} \in \{0,1\}$$

$$\forall i \in F: \qquad y_i \in \{0,1\}$$

F is a set of facilities and C is a set of all nodes in network. c_{ij} denotes the linear cost function of connecting node j to the (opened) facility i, called connection cost. y_i and x_{ij} are the decision variables. y_i equals to 1 if and only if a facility i is located (or selected). x_{ij} equals to 1 if and only if node j is served by the facility i. We will consider the metric version of this problem, that is, the assignment costs are non-negative, symmetric, and satisfy the triangle inequality: that is, $c_{ij} = c_{ji}$ and $c_{ij} + c_{jk} \geq c_{ik}$.

The problem is to locate the facilities such that the total connection cost is minimized. The first constraint ensures that each node is connected to at least one facility (if exactly one facility is allowed to be connected to, the inequality must be replaced by the equal mark), and the second ensures that this facility must be open. The third constraint keeps the number of facilities lower than or equal to *k*.

k-median and *k*-center are similar in that both problems take the number of clusters *k* as an input. Furthermore, the second and the third constraints of the two problems are identical. Nevertheless, it is important to emphasize a major different between *k*-center and *k*-median objective function. *k*-center problem minimizes the maximum cluster size while *k*-median

minimizes total connection cost, that is, the total distances within clusters. At this point, we might say that *k*-median is a stronger optimization model since it concerns all the distances that connect each node to the nearest facility, while *k*-center problem concerns only on the maximum cluster size. The average distance between a node and the nearest facility of *k*-median model is lower than or equal to that of *k*-center model.

The *k*-median formulation given above assumes that facilities are located on nodes, which can be compared to a vertex *k*-center problem. This can lead to suboptimal solutions, as we have discussed earlier. However, [HAKI65] has shown that for the *k*-median problem, at least one optimal solution consists of locating *k* facilities on the network's nodes. A brief proof can be found in [DASK95].

Solving *k*-median on a tree can be done deterministically in polynomial time. [GOLD71] gave an $O(n)$, a linear time, algorithm for solving *1*-median problem on a tree. [KARI79] provided an $O(n^2k^2)$ algorithm for finding *k* medians on a tree with *n* nodes. The problem becomes NP-complete on a general graph [GARE79]. Numerous approximation algorithms have been proposed for many years. The breakthrough was made by [CHAR99a] who gave a $6\frac{2}{3}$-approximation algorithm, the first constant factor approximation algorithm for the *k*-median problem. The algorithm, however, has a prohibitive running time since it is based on LP-rounding, which means it needs to solve large integer linear programs. [JAIN99] improved to factor of 6, using the Primal-Dual technique. In the very same year, [CHAR99b] gave a 4-approximation algorithm, based on the idea of [JAIN99], combining with greedy augmentation, cost scaling, and LP-based algorithm. The algorithm runs in $\tilde{O}(n^3)$ time. Recently, [ARYA01] proposed a local search heuristic with multiple swaps that yields the factor of 3+2/*p*, and running time $O(n^p)$, where *p* is the number of facilities that are allowed to be swapped in each round. The local search algorithm in [ARYA01] is outlined as follows:

1. $\boldsymbol{S}$:= an arbitrary feasible solution.
2. While there exists an operation $swap(\boldsymbol{S})$ such that,
$$cost(swap(\boldsymbol{S})) \leq (1 - \frac{\varepsilon}{p(k,n)})cost(\boldsymbol{S}),$$ do
$\boldsymbol{S} \leftarrow swap(\boldsymbol{S})$;
end while
3. Return $\boldsymbol{S}$.

Here $\varepsilon > 0$ is a constant, $k = |F|$ is the number of facilities, $n = |C|$ is the number of nodes and $p(k, n)$ is a polynomial in k and n. The cost function $cost(\boldsymbol{S})$ returns the value of the objective function of k-median problem. A $swap(\boldsymbol{S})$ is affected by closing a facility $s \in \boldsymbol{S}$ and opening a facility $s' \notin \boldsymbol{S}$. Mathematically, the operation $swap(\boldsymbol{S})$ is defined as,

$$swap(\boldsymbol{S}) = \boldsymbol{S} - s + s' \qquad \text{for } s \in \boldsymbol{S} \text{ and } s' \notin \boldsymbol{S}.$$

An operation $swap(\boldsymbol{S})$ is called admissible for $\boldsymbol{S}$ if $cost(swap(\boldsymbol{S})) \leq (1 - \varepsilon / p(k, n))cost(\boldsymbol{S})$. During each admissible $swap(\boldsymbol{S})$, the cost of the current solution decreases by a factor of at least $\varepsilon / p(k, n)$. At any execution of the step 2 of the algorithm, there will be a most a polynomial number of $swap(\boldsymbol{S})$ to be checked for admissibility, i.e., there are $k \times (n - k) = nk - k^2$ operations to be checked. If $\boldsymbol{S}^*$ denotes an optimum solution and $\boldsymbol{S_0}$ denotes the initial solution, then the number of *swap* that the algorithm does is at most $log(cost(\boldsymbol{S_0})/cost(\boldsymbol{S}^*))/log\frac{1}{1-\varepsilon / p(k,n)}$. The proof can be found in [ARYA01]. As $log(cost(\boldsymbol{S_0}))$ is polynomial in the input size and performing each *swap* takes a polynomial time, this algorithm terminates in polynomial time.

An arbitrary feasible solution, $\boldsymbol{S_0}$, in step 1 of the algorithm can be randomized. However, a small $cost(\boldsymbol{S_0})$ is desirable since it will reduce the number of *swap* operations and, thus, reduce running time of the algorithm.

[DASK95] outlined two classes of approximation algorithms for general discrete location problems: *construction algorithms* and *improvement algorithms*. For example, finding an arbitrary feasible solution, $\boldsymbol{S_0}$, in step 1 of the algorithm above can be done by using one of the construction algorithms in which the solution is built from scratch. Step 2 is considered an improvement algorithm in which the solution is based on the improvement of the previous solution. [DASK95] provided a construction algorithm, called myopic algorithm, which can be described as follows:

1. $\boldsymbol{S} := \emptyset$
2. While $|\boldsymbol{S}| < k$ do
   find the location I that minimizes
   $$Z_I := \sum_{j \in C} \min_{i \in S \cup I} c_{ij} \qquad : \forall \, I \in F \text{ and } I \notin \boldsymbol{S};$$
   $\boldsymbol{S} = \boldsymbol{S} \cup I$;
   end while
3. Return $\boldsymbol{S}$

The solution, $\boldsymbol{S}$, is a set of located (or selected) facilities. The algorithm above terminates when the number of located facilities reaches k. c_{ij} is a cost (or distance) of connecting node j to the facility i. The *min* function ensures that the smallest c_{ij} is selected. That is, node j connects to the nearest facility $i \in \boldsymbol{S} \cup I$. The algorithm tries every facility $I \notin \boldsymbol{S}$ and adds facility I, which yields the smallest Z_I, to the solution $\boldsymbol{S}$.

Despite the fact that the solution may not be optimal, this algorithm is appealing for its simplicity. Besides, this method of construction the solution gives us the knowledge of the solution to the *1*-median through *(k-1)*-median. Hence, we can stop at any number k, if the solution satisfied our criteria, e.g. the target coverage distance is reached. This is a significant feature as we will see later. It is easy to see that the myopic approach presented above is a greedy adding algorithm. The location that yields the smallest objective value is added to the set of solution at each round. Also, the located facility cannot be removed or exchanged later with other facility. The algorithm runs in $O(kn^2)$ time, a polynomial time. However, there is no approximation factor guaranteed.

2.3.1.4 The Uncapacitated Facility Location Problem

The three discrete location models discussed earlier require either the number of facilities to be located or the maximum allowable coverage distance as an input. For example, k-median and k-center problems take the number of facilities, k, as an input, while set covering problem takes coverage distance, d_c, as an input. To illustrate the necessity of these inputs, consider Figure 2-3. The figure shows, in the case of 88-node k-median problem [DASK95], the average distance as a function of the number of facilities that are located. The figure suggests that we should locate as many facilities as possible, in order to reduce the average distance between a demand node and

the nearest facility. Therefore, the number of facilities must be limited at some point; otherwise all the facilities will be located.

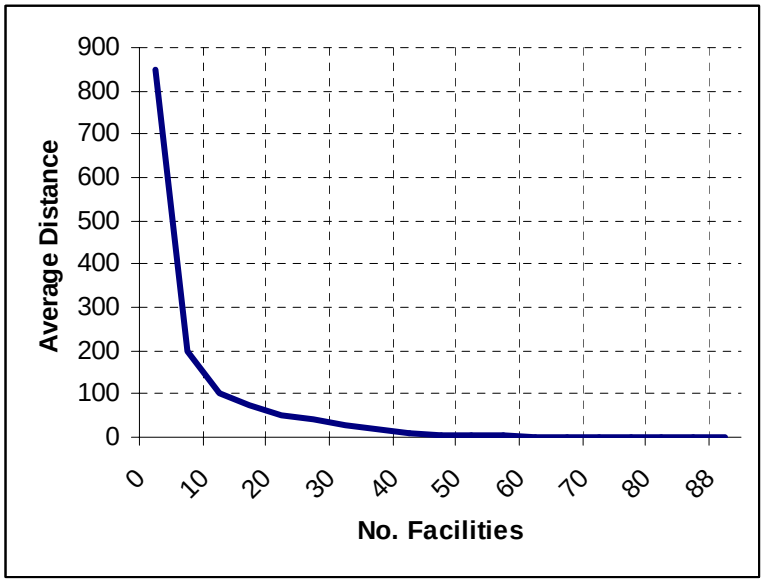

Figure 2-3: Average distance versus number of facilities for the 88-node *k*-median problem.

In this section, we introduce the uncapacitated facility location problem (UFLP), which neither parameter k nor d_c is required as an input. Other than the connection cost (a.k.a. service cost), the model incorporates the cost of locating facilities into the objective function. The cost of locating facilities is sometimes called facility opening cost. The uncapacitated facility location problem minimizes the total cost – the sum of the connection cost and the facility opening cost. The connection cost and the facility opening cost must be in the same unit; usually, the monetary unit is used. For example, if we were to locate bank branches, the facility opening cost is the cost of building each bank branch, and the connection cost is the cost per mile that each customer has to pay for traveling to the nearest bank branch. In this problem we wish to locate bank branches so that all customers are covered within the smallest transportation cost, yet minimum budget is spent for building bank branches.

The uncapacitated facility location problem determines both the number of facilities to be located and the maximum coverage distance endogenously. This can be described by Figure 2-4.

Figure 2-4 replicates Figure 2-3 but adds a constant fixed cost for each facility that is located. In addition, the vertical axis is changed to cost assuming that the cost is in the same scale as average distance, i.e. multiply by 1. The sum of facility opening cost and connection cost yields total cost. As the number of located facilities increases, the total cost initially declines because of the reduction in connection cost. At some point, the facility opening cost dominates the connection cost and causes total cost increases as we add more facilities. For the problem in Figure 2-4, the optimal number of facilities is 10 and optimal cost is around 200.

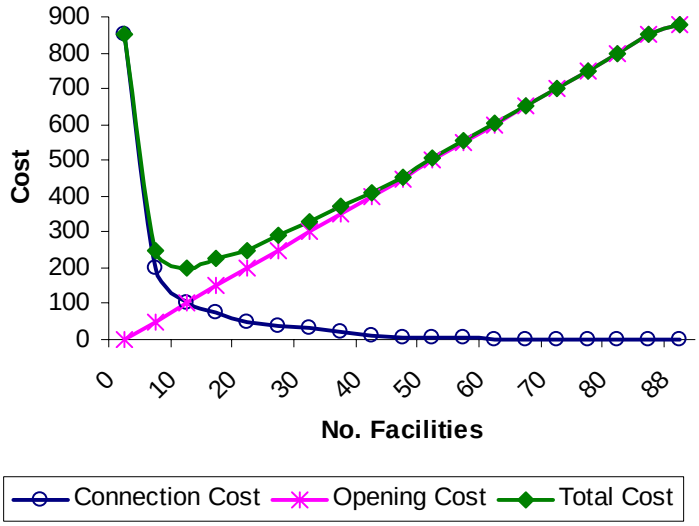

Figure 2-4: Cost versus number of facilities for the 88-node UFLP.

The uncapacitated facility location problem can be mathematically defined as follows. Let F be a set of facilities, and C be a set of cities (i.e., nodes in the networks). For every facility $i \in F$, a nonnegative number f_i is given as an opening cost. Furthermore, for every facility $i \in F$ and city $j \in C$, we have a connection cost c_{ij} between facility i and city j. The objective is to open a subset of the facilities in F, and connect each city to an open facility so that the total cost is minimized. We, again, consider the metric version of this problem. The term uncapacitated means each facility has unlimited capacity for serving the cities. Note that the k-median problem can be

viewed as a special case of uncapacitated facility location problem, where all facility opening costs are set to zero and maximum of k facilities are allowed to be opened.

The metric uncapacitated facility location problem can be modeled as an integer linear program as follows [JAIN99]:

$$\textbf{MINIMIZE} \qquad \sum_{i \in F, j \in C} c_{ij} x_{ij} + \sum_{i \in F} f_i y_i$$

$$\begin{aligned}
\textbf{SUBJECT TO:} \quad & \forall j \in C: && \sum_{i \in F} x_{ij} \geq 1 \\
& \forall i \in F, j \in C: && y_i - x_{ij} \geq 0 \\
& \forall i \in F, j \in C: && x_{ij} \in \{0,1\} \\
& \forall i \in F: && y_i \in \{0,1\}
\end{aligned}$$

In this program, y_i is a decision variable denoting whether facility i is open, and x_{ij} is a decision variable denoting whether city j is connected to the facility i. The first constraint ensures that each city is connected to at least one facility, and the second constraint ensures that this facility must be open. The program can be relaxed to the canonical form by replacing the third and fourth constraints by the nonnegativity constraints. Also note that, sometimes, the first constraint can be expressed in the form of equality to ensure that each city connects to only one facility (e.g. see [MIRC90]).

The facility location problem and its variations have been proved NP-complete [GARE79]. Particularly, the uncapacitated facility location is NP-hard. The proof can be found in [MIRC90]. Number of approximation algorithms has been proposed in the literature.

[KUEH63] was one of the earliest to introduce the fundamental heuristics for solving general discrete location problems. Their approach consists of two routines. The first routine is used to establish an initial solution by opening facilities sequentially in an order that maximizes the decrease of the objective function value at each step. It stops when opening a new facility would increase the value. This routine is known as the *greedy improvement heuristic* because of its appetite for maximum improvement at each step. This is equivalent to the construction algorithms discussed earlier. Their second routine is referred as an *interchange heuristic*. It considers interchanging an open facility with a closed facility. Such a pairwise interchange is

performed if it improves the current feasible solution and the procedure stops when a solution cannot be further improved by such interchanges. This is similar to the local search technique with swaps proposed by [ARYA01] for solving *k*-median problems. The interchange heuristics are considered the improvement algorithms discussed earlier. The greedy improvement and interchange heuristics are the basis of numerous approximation algorithms.

The first approximation algorithm for the facility location problem, a greedy algorithm achieving a guarantee factor of $O(\log n)$ proposed by [HOCH82], dates back to 1982, more than 20 years ago. Recent works have shown dramatic improvements. The first constant factor approximation algorithm for this problem was given by [SHMO97]. [JAIN99] gave a primal-dual algorithm, achieving approximation factor of 3 and running time of $O(m \log m)$, where m is the number of edges presented in the underlying graph. (We will refer to this as the JV algorithm.) The importance of the JV algorithm is that it is fundamental to the many succeeding algorithms. It was also adapted for solving several related problems such as the fault-tolerant and outlier versions (see [JAIN00] and [CHAR01]). [JAIN02] and [JAIN03] improved the JV algorithm and yielded the approximation factor of 1.61, running time of $O(n^3)$. [THOR03] gave a quick randomized algorithm that yielded, with high probability, approximation factor of 1.62 in $\tilde{O}(n + m)$ time. Here $\tilde{O}$ means that we suppress logarithmic factors. These two results are very close to the best possible approximation factor in that [GUHA98] have shown that we cannot get an approximation factor below 1.463 in polynomial time unless $NP \subseteq DTIME[n^{O(\log \log n)}]$. [MAHD02] combined the idea of [JAIN03] with the cost scaling technique, achieving approximation factor of 1.52, which is the best approximation factor known for this problem.

The algorithm in [JAIN03] is vital to understand since it is the fundamental to the recently best known algorithms in both term of running time [THOR03] and accuracy [MAHD02]. The algorithm can be described as follows. The network consists of facilities $i \in F$ and cities $j \in C$. U is a set of unconnected cities. The algorithm introduces a notation of time, so that each event can be associated with the time at which it happened. The algorithm starts at time 0. At this time, each city is defined to be unconnected ($U := C$), all facilities are unopened, and contribution α_j is set to 0 for every j.

At every moment, each city j offers some money from its contribution to each unopened facility i. The amount of this offer is computed as follows: If j is unconnected, the offer is equal to $\max(\alpha_j - c_{ij}, 0)$ (i.e., if the contribution of j is more than the cost that it has to pay to get

connected to i, it offers to pay this extra amount to i); If j is already connected to some other facility i', then its offer to facility i is equal to $max(c_{i'j} - c_{ij}, 0)$ (i.e., the amount that j offers to pay to i is equal to the amount j would save by switching its facility from i' to i).

While $U \neq \emptyset$, increase the time, and simultaneously, for every city $j \in U$, increase the parameter α_j at the same rate, until one of the following events occurs (if two events occur at the same time, we process them in an arbitrary order).

(a) For some unopened facility i, the total offer that it receives from cities is equal to the cost of opening i. In this case, we open facility i, and for every city j (connected or unconnected) which has a nonzero offer to i, we connect j to i. The amount that j had offered to i is now called the contribution of j toward i, and j is no longer allowed to decrease this contribution.

(b) For some unconnected city j, and some open facility i, $\alpha_j = c_{ij}$. In this case, connect city j to facility i and remove j from U.

Note that the above algorithm makes greedy choices in deciding which facilities to open and once it opens a facility, it does not alter this decision. In this sense, it is also a greedy algorithm.

2.3.2 Discussion on the discrete location mode

Recently, the discrete location problems have been extensively studied from the perspective of approximation algorithms. As we have seen, the approximation factor of k-median problem has been developed from the first constant factor of $6\frac{2}{3}$ to $3 + 2/p$ within a few years. Likewise, in facility location problem, the best known approximation factor is 1.52, improved from the first constant factor of 3, recently. However, from the practitioners' point of view, these approximation factors allow too high error. For example, approximation factor of 1.52 means the error can be as high as 52% from the optimal, in the worst case.

Fortunately, in practice, the algorithms work much better than the guaranteed approximation ratios. [GROS97] studied several algorithms and showed that a simple greedy algorithm gives satisfied results for solving the set covering problems. For UFLP, [JAIN02] and [JAIN03]

implemented their works on several randomly generated test cases and found that the solution given by their algorithms was at most a factor of 1.05 away from the lower bound obtained by solving the integer linear programming relaxation (which is assumed to be an optimum) of the problem. For the case of *k*-median problem, we tested a simple greedy algorithm against 40 problems from [ORLIB], where the optimal solutions are known. The worst case error is less than 5% from optimum. The detailed analysis on the performance of the algorithms in practice and the experimental results are given in the next chapter.

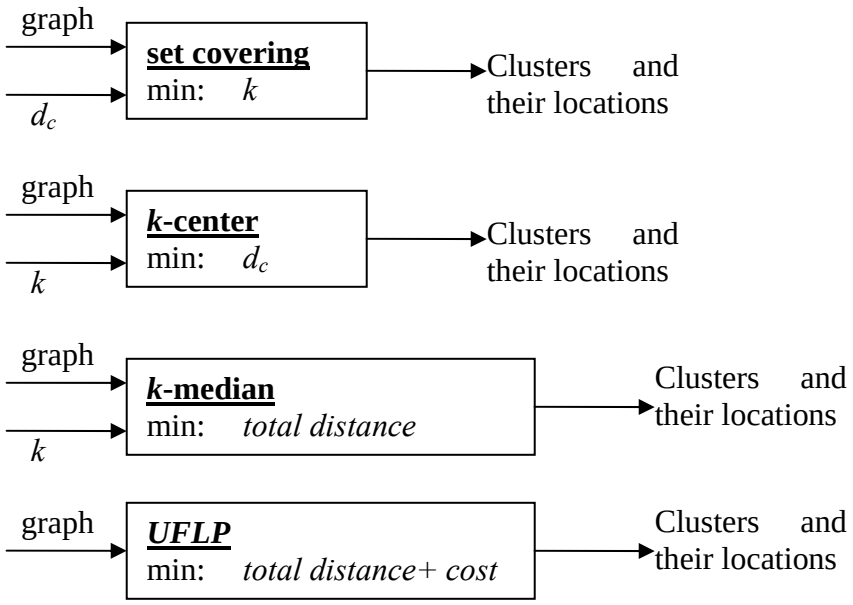

Figure 2-5: Summary of the four discrete location models.

The characteristics of the four discrete location models are summarized in Figure 2-5. The objective of the discrete location models is to optimally cluster the network. Hence, the outputs of every model are the clusters and their locations. The set covering problem takes the coverage distance, d_c, as an input and tries to minimize the number of clusters, k. Conversely, the k-center problem takes k as an input and tries to minimize d_c. Set covering model guarantees that the maximum diameter of the clusters is not larger than the coverage distance d_c. Similarly, k-center model guarantees that the total number of clusters is not more than k. It's easy to see that the two problems are dual to each other. The duality is in the sense that, given a graph and a constant d_c,

we obtain k from solving set covering problem. This k, applying to k-center problem, yields the exact d_c. And, the both solutions give the same clustering results. Since the two models are equivalent and selecting between models is analogous to selecting types of inputs. At this point, we suggest using the set covering model for our selective probing scheme since the coverage distance d_c makes more sense to the network administrator than the number of cluster k.

k-median, on the other hand, cannot guarantee the maximum cluster size. The solution to the k-median problem tends to have more varied cluster sizes, compared to the solution of set covering or k-center problem. This is because the objective function of k-median focuses on minimizing the distances within clusters, regardless of the cluster size. The benefit of this approach, however, is that the average distance from a node to the nearest facility (anchor) is minimized, which is desirable.

UFLP seems to be the most appealing model since no input parameter is needed. UFLP, however, requires the facility opening cost to be specified in the same scale of the connection cost. Usually, in general facility location problems such as locating bank branches, the monetary unit are used, e.g. using the cost of building bank branch as a facility opening cost, and customer's transportation cost as a connection cost. In our selective probing scheme, however, the connection cost is presented by delay. Therefore, it's difficult to specify the corresponding facility opening costs. To illustrate the problem, consider again Figure 2-4. If a constant facility opening cost is used for every facility, the higher cost determines a steeper slope. Thus, changing facility opening cost affects the total cost. Therefore, the optimal solution of UFLP depends on a constant assigned as the facility opening cost.

Despite of the fact that the practical use of UFLP is prohibited for our selective probing scheme, it can be adapted to solve k-median problem. Clearly, if the facility opening cost = 0, the algorithm will open all facilities, and if the facility opening cost is very high, it will open only one facility. We then perform a binary search on the opening costs for the instance of UFLP that yields minimum objective value and opens the number of facilities less than or equal to k. The technique is similar to the one presented in [JAIN99]. We have tried [JAIN03] algorithm (for UFLP) solving the same problem set as for k-median [ORLIB], using the above technique. Unfortunately, the solutions can be deviated as high as 40% from the optimum, in the worst case. Details on the experimental results will be elaborated in the next chapter. We conclude that UFLP is not suitable for our selective probing scheme.

2.4 Relevant works

The discrete location models are the central of studies in operational research. [MIRC90] and [DASK95] provided good surveys on the applications to the discrete location models. Recently, they are used in network design problems such as placement of routers and caches [LI99], [GUHA00] and web server replications in a content distribution network (CDN) [JAMI00], [QIU01]. Discrete location models are also presented in a network distance monitoring and estimation systems such as the work proposed by [CHEN02].

[QIU01] described how to locate web server replicas in the content distribution networks by solving the *k*-median problem. This work is very similar to ours in the sense that it used the discrete location models to find the near optimal solution (using variation of greedy approximation algorithms) of locating web server replicas. Essentially, the work focuses on comparing the performance of a set of algorithms. The performance of an algorithm determines by the ratio of the cost of the solution found by the algorithm to the cost determined by the optimal solution. The algorithms used in this work are tree-based, greedy, random, hot spot algorithm. The tree-based algorithm, as the name suggests, works well only on a tree and not a general graph. The greedy algorithm is similar to ours. The random algorithm randomly chooses *k* web server replicas. The hot spot algorithm attempts to place *k* web server replicas near the clients generating the greatest load. Beside the fact that these algorithms have no approximation factor guaranteed, they are poorly invented and this make the comparison uninteresting.

[CHEN02] proposed the overlay network distance monitoring and estimation system called the Internet Iso-bar. This work divides an overlay network into clusters and estimates the distance (delay) between any pair of nodes by inferring using both distance between clusters and distance within clusters. The Internet Iso-bar uses set covering model and the greedy approximation algorithm suggested by [GROS97], which is similar to ours.

2.5 Conclusion

In this chapter, we have discussed the overview of the QoS routing algorithm and protocol. The QoS routing is essential for supporting multimedia applications; however, it turns out to be an intractable problem. Therefore, many heuristics and approximation algorithms have been proposed in the literature. The QoS routing protocol, on the other hand, has to deal with the tradeoff between scalability and accuracy. We proposed the clustering technique and selective probing to address this tradeoff.

We then discussed the power laws of the Internet and small-world phenomena that capture the static properties of the Internet topology. These properties have been used by the new Internet topology generators for generating the more realistic models. However, the weights information on the links of the generated graphs is missing and is impossible to obtain. We overcome this by proposing the sensitivity analysis for our study.

For the clustering technique, four discrete location models have been discussed, which are set covering model, *k*-center model, *k*-median model, and uncapacitated facility location model. These models are in the form of optimization problems, which differ in both objective functions and constraints. Every model has been proved intractable; and the approximation algorithms were given. We concluded that among the discrete location models discussed so far, only set covering is suitable for our work.

Next chapter, we will propose a dual to *k*-median problem, called *d*-median problem. Then the evaluation of both the set covering and *d*-median models is given. We will investigate and compare their behavior when used them to cluster the Internet. We also try to find the practical range of the input parameters for each model.

3.0 THE *d*-MEDIAN CLUSTERING APPROACH: DESIGN AND ANALYSIS

In this chapter, we start with introducing the *d*-median clustering method. Then, we study in depth the behavior of the two clustering methods; the set covering and *d*-median, on the synthesized power-law graphs. We will investigate the behavior of the two approaches and analyze the properties that are beneficial to our selective probing scheme. In particular, we will be focusing on the *d*-median clustering method, which outperforms set covering method in many cases, as we will see later.

The rest of this chapter is organized as follows. We start by defining the *d*-median clustering method. Then we proceed to evaluation methodology, which gives the definitions and scopes of the variables, parameters and environments of the experiments. We describe the two greedy algorithms in details by examples and define the performance parameters, which we will use as the tools for our study. Then, we show the experimental results and analysis. Finally, we conclude the chapter.

3.1 The *d*-median clustering method

As it has been discussed in the last chapter that *k*-median has a desirable property in that it tries to minimize the delay (distance) between every node and its nearest anchor. However, the model cannot guarantee the maximum delay bound from an anchor to the farthest node in its cluster. We need an alternative model that behaves like *k*-median, yet guarantees the maximum delay bound. In other words, the model must combine the advantages of both set covering and *k*-median approaches. Such model can be achieved by modifying the constraints of *k*-median problem so that the maximum delay of each cluster is bounded. This model takes the coverage

distance (maximum delay bound), d_c, as an input and determines the number of clusters, k. We call this model d-median approach.

The d-median approach tries to locate minimum number of anchors such that the sum of the connection cost is minimize and the maximum delay of every cluster does not exceed the delay bound input, d_c. The d-median problem may be formulated in a form of optimization problem as follows:

$$\textbf{MINIMIZE} \qquad \sum_{i \in F, j \in C} c_{ij} x_{ij}$$

$$\begin{aligned}
\textbf{SUBJECT TO:} \quad & \forall j \in C: && \sum_{i \in F} x_{ij} \geq 1 \\
& \forall i \in F, j \in C: && y_i - x_{ij} \geq 0 \\
& \forall j \in C: && D_c \geq c_{ij} x_{ij} \\
& \forall i \in F, j \in C: && x_{ij} \in \{0,1\} \\
& \forall i \in F: && y_i \in \{0,1\}
\end{aligned}$$

Again, F is a set of anchors and C is a set of all nodes in network. c_{ij} denotes the connecting cost of node j and anchor i. y_i and x_{ij} are the decision variables. y_i equals to 1 if and only if an anchor i is located (or selected). x_{ij} equals to 1 if and only if node j is served by the anchor i. Basically, the objective functions of d-median and k-median are identical. The only difference is the third constraint, where it says that the coverage distance of each cluster must not exceed d_c, which bound maximum delay within the clusters to d_c.

We can also modify the greedy construction algorithm of k-median for the case of d-median as follows:

1. $\boldsymbol{S} := \emptyset$
2. While there exist node j, which connection cost to the nearest facility exceeds d_c, do
   find the location I that minimizes
   $$Z_I := \sum_{j \in C} \min_{i \in S \cup I} c_{ij} \quad : \forall \, \mathrm{I} \in F \text{ and } \mathrm{I} \notin \boldsymbol{S}$$
   $\boldsymbol{S} := \boldsymbol{S} \cup I$
   end while
3. Return $\boldsymbol{S}$.

The algorithm greedily selects the anchor that yields the smallest total connection cost, one-by-one. The *min* function ensures that every node connects to its nearest anchor. The algorithm repeats until every node in the network has been assigned the anchor with the connection cost that does not exceed the delay bound input, d_c. The solution to the algorithm, $\boldsymbol{S}$, is a set of selected anchors. Each anchor and their members then form a cluster. The running time of this algorithm is $O(n^3)$.

3.2 Evaluation methodology

Performance evaluation of the *d*-median and set covering clustering methods is done by observing their clustering results on the various synthetic Internet topologies. Since the Internet grows larger every day and the delay distribution of the links in the Internet is not yet known, we perform the sensitivity analysis on the network sizes and the delay distributions. The sensitivity analysis is done based on the hypothesis that size and delay distribution of the Internet do not affect the behavior of clustering methods, as long as the topology conforms to power laws.

Furthermore, from the sensitivity analysis on the power-law graph, we will try to decide the suitable range of the delay bound input (or coverage distance) for the two clustering methods. Although the delay bound is usually decided by applications or administrators, it is beneficial to know roughly the range of efficient delay bound because a too-large delay bound causes high amount of estimation errors and the too-small delay bound causes the scheme not scalable.

In brief, our objectives of the study are as follows. First, we study the behavior of set covering and *d*-median clustering method on the power-law graphs. Second, we perform the sensitivity analysis on the network sizes and delay distributions, given that the network is power-law graph. Finally, we investigate the potential range of the delay bound inputs.

In this section, we will first discuss about the Internet topology generation. Then, we give the details on the two clustering methods and provide two examples on their greedy algorithms. Finally we define the performance metrics that will be used to evaluate their performance.

3.2.1 The Internet topology

As we have pointed out in chapter 2.0 that the measured Internet topology is impractical for this early stage of study, unless the technique for summarizing the input data is developed; the synthesized power-law graphs are preferable, as long as the Internet topology conforms to power laws. Among many Internet topology generators existing in the literature, we choose INET 3.0. The features of INET 3.0 have been summarized in [WINI03]. Many metrics were developed for validating the topology generator, including degree distributions, derived from power laws of the Internet [FALO99], and the characteristic path length, derived from the small-world phenomena [WATT98]. Still, the two criteria that INET 3.0 has not yet satisfied are the maximum clique size and the clustering coefficient.

Neither INET 3.0 generated graphs nor the measured Internet topology supplies with the knowledge of the bandwidth or delay on the links; only the connectivity information is presented. However, in our delay-based selective probing scheme, the knowledge of the delays on the links is required. We cannot presume any statistical distribution for representing the delay on the Internet links. Therefore, we conduct the sensitivity analysis among four well-known statistical distributions; namely, uniformed distribution, normal distribution, exponential distribution and power-law (heavy-tailed) distribution.

The observation in the early stage of the study of the power laws of the Internet showed that there exist 3,037 nodes in the Internet AS-level topology, as of in the year 1997. Therefore, the size of our generated network topologies must not be smaller than 3,037 nodes. In addition, we also do the sensitivity analysis on the size of the network using four network size instances; 3,037 nodes, 3,500 nodes, 4,000 nodes and 4,500 nodes. Although, today's Internet topology

[CAIDA] consists of more than 200,000 nodes, power laws still hold. Thus, our experiments are done on the same class of graphs to the real Internet topology, the power-law graphs, but in the smaller size.

3.2.2 Clustering methods

This work focuses on the methods of clustering the network so that the QoS measurement can be done efficiently. Among four discrete location problems discussed in chapter 2.0, we will be using only the set covering problem and the *k*-median problem for our selective probing scheme. Moreover, the *k*-median problem has been developed so that the delay can be used as an input, called *d*-median problem.

Although the best known approximation factors for discrete location algorithms allow the solutions to deviate as high as 60% to 400% from the optimum, the experiments showed that the errors are quite small in practice, using simple greedy-based algorithms. A simple greedy-based approximation algorithm for the set covering problem was tested by [GROS97] against 60 random problems from [ORLIB], comparing with other algorithms. The performance of the greedy-based algorithms outperforms other algorithms in both terms of running time and accuracy. On average, the greedy-based algorithms deviate merely 5% from optimum.

For *k*-median problem, we run a simple greedy construction algorithm against 40 test cases from [ORLIB]. The results are shown in Table 3-1. The average deviation from the optimum is very low, 1.51%, while the worst case error is only 4.3%. As we have addressed earlier, the UFLP cannot be used directly unless the appropriate facility opening costs are known beforehand. Table 3-1 also shows the results of using UFLP for solving *k*-median problem by performing a binary search on the opening costs for the instance of UFLP that yields minimum objective value and opens the number of facilities less than or equal to *k*. The running time of the algorithm is higher than solving *k*-median problem using greedy algorithm since it requires solving many instances of UFLP during the binary search. Besides, both average deviation from the optimum and the worst case error are very high, which are 6.82% and 39.36%, respectively. In conclusion, the direct and indirect usages of UFLP are considered unsuitable for our selective probing scheme. We will be considering only the set covering problem and *k*-median problem in our work; whereas the modified version of *k*-median called *d*-median will be used.

Table 3-1: Relative error of greedy k-median and greedy UFLP algorithms.

Problem Instances	*n*	*k*	*% k / n*	*OPT SOL*	*k-MEDIAN*		*UFLP*	
					SOL	*%ERROR*	*SOL*	*%ERROR*
pmed01	100	5	5.00	5819	5891	1.24	5891	1.24
pmed02	100	10	10.00	4093	4118	0.61	4128	0.86
pmed03	100	10	10.00	4250	4399	3.51	4331	1.91
pmed04	100	20	20.00	3034	3088	1.78	3215	5.97
pmed05	100	33	33.00	1355	1378	1.70	1779	31.29
pmed06	200	5	2.50	7824	8027	2.59	7897	0.93
pmed07	200	10	5.00	5631	5646	0.27	5646	0.27
pmed08	200	20	10.00	4445	4472	0.61	4475	0.67
pmed09	200	40	20.00	2734	2841	3.91	2978	8.92
pmed10	200	67	33.50	1255	1295	3.19	1749	39.36
pmed11	300	5	1.67	7696	7721	0.32	7714	0.23
pmed12	300	10	3.33	6634	6651	0.26	6659	0.38
pmed13	300	30	10.00	4374	4467	2.13	4538	3.75
pmed14	300	60	20.00	2968	3013	1.52	3225	8.66
pmed15	300	100	33.33	1729	1761	1.85	2168	25.39
pmed16	400	5	1.25	8162	8232	0.86	8266	1.27
pmed17	400	10	2.50	6999	7019	0.29	7061	0.89
pmed18	400	40	10.00	4809	4873	1.33	4987	3.70
pmed19	400	80	20.00	2845	2899	1.90	3009	5.76
pmed20	400	133	33.25	1789	1866	4.30	2237	25.04
pmed21	500	5	1.00	9138	9138	0.00	9138	0.00
pmed22	500	10	2.00	8579	8670	1.06	8723	1.68
pmed23	500	50	10.00	4619	4694	1.62	4757	2.99
pmed24	500	100	20.00	2961	3009	1.62	3163	6.82
pmed25	500	167	33.40	1828	1896	3.72	2348	28.45
pmed26	600	5	0.83	9917	10093	1.77	10095	1.79
pmed27	600	10	1.67	8307	8364	0.69	8410	1.24
pmed28	600	60	10.00	4498	4579	1.80	4759	5.80
pmed29	600	120	20.00	3033	3104	2.34	3308	9.07
pmed30	600	200	33.33	1989	2037	2.41	2572	29.31
pmed31	700	5	0.71	10086	10086	0.00	10086	0.00
pmed32	700	10	1.43	9297	9331	0.37	9326	0.31
pmed33	700	70	10.00	4700	4798	2.09	4791	1.94
pmed34	700	140	20.00	3013	3097	2.79	3271	8.56
pmed35	800	5	0.63	10400	10406	0.06	10406	0.06
pmed36	800	10	1.25	9934	9954	0.20	9999	0.65
pmed37	800	80	10.00	5057	5118	1.21	5216	3.14
pmed38	900	5	0.56	11060	11153	0.84	11086	0.24
pmed39	900	10	1.11	9423	9451	0.30	9449	0.28
pmed40	900	90	10.00	5128	5190	1.21	5335	4.04
AVG						1.51		6.82

Note that, in our selective probing scheme, the perfect solution is not necessarily required, although the near optimal solution is desired to minimize the number of cluster *k*. The experiments on the performance of the greedy algorithms on the set of random problems indicate that greedy algorithms perform sufficiently well in practice. Additionally, the simplicity of the greedy-based algorithms has the advantage in the fast running time, which is preferable for the large scale network.

From the experimental results, we observe that the error from the approximation algorithms tends to be higher when the ratio *k/n* (clusters per total nodes) is large. We plot the relationship between error and the ratio *k/n* in Figure 3-1. This suggests that the approximation algorithms are likely to perform well when $k << n$. We will see later that the ratio *k/n* we obtain in our selective probing scheme is very small, e.g., 0.005 – 0.15, which implies that the clustering results are near optimal in most cases.

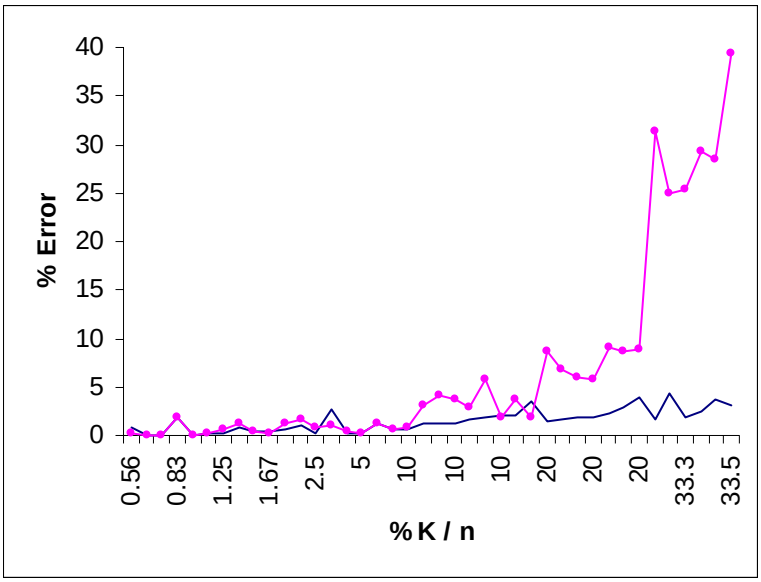

Figure 3-1: The ratio k/n versus relative error.

3.2.3 The greedy algorithms

In this section, we describe in details the greedy algorithms for the set covering and *d*-median problem that will be evaluated in this work.

3.2.3.1 Greedy set covering algorithm

For set covering problem, the greedy algorithm is straightforward as outlined earlier. Each node is treated as a potential facility and the nodes within its coverage distance are the members of the cluster. At each round, the algorithm greedily adds a cluster that covers most of the nodes in the network to the set of solution. In the case of tie, the cluster with the smallest connection cost and the highest out-degree facility is selected, respectively. The smallest connection cost cluster is chosen in favor of minimizing the total connection cost, as in the case of *k*-median problem. The out-degree reflects the importance of the node. Then, the nodes in the selected cluster, i.e., the facility and its members are marked covered. The algorithm stops when the all the nodes in network are marked covered.

To illustrate the greedy set covering algorithm, consider the network in Figure 3-2 (adapted from [DASK95]). The labels on the links represent delay. The network consists of 12 nodes and 21 links with the delay ranging from 12 to 30. The Figure 3-2 gives the distance matrix, which may be found by using Floyd's algorithm for the shortest paths from all sources to all destinations. In the case of undirected graph, we obtain a symmetric distance matrix.

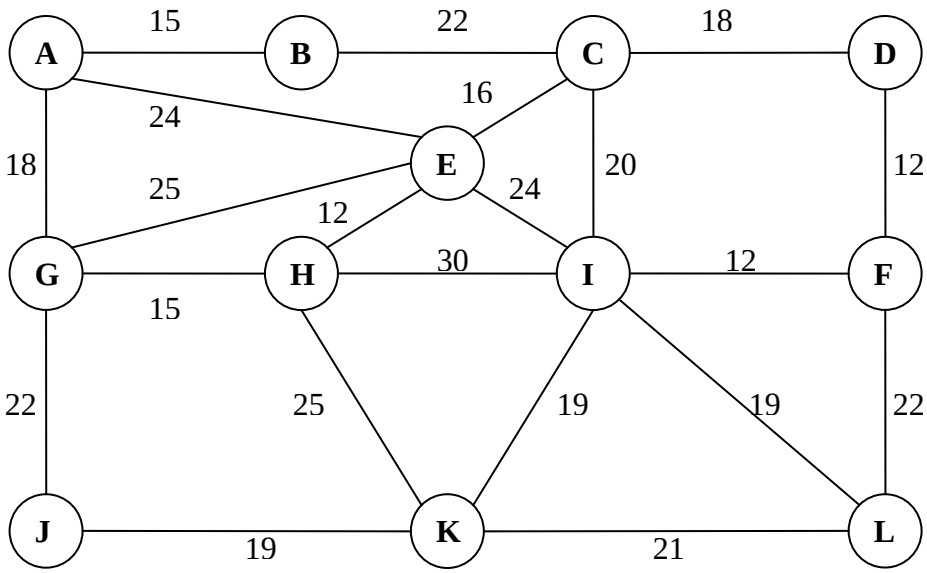

Figure 3-2: An example network for greedy set covering problem.

Suppose the coverage distance d_c is 20. For each node, we count the number of its members, including itself, that located within the coverage distance. We find that *I* covers the most number of the nodes in the network, 5 nodes, namely *C, F, I, K* and *L*. Node *I* and its members form a cluster and are marked covered, excluding them from further calculation. We then consider the rest of the nodes in the network and find that *A, G* and *H* tie in the number of nodes covered; each covers 3 nodes. Comparing between the connection costs, the cluster of node *H* yields the smallest connection costs of 27, *H-G* and *H-E*. Node *H* and its members, again, form a cluster and marked covered. The algorithm continues in this manner until all the nodes in the network are marked covered. Figure 3-3 illustrates the resulting clusters of the network in Figure 3-2. The algorithm partitions the network into 5 clusters with the largest cluster containing 5 nodes and the smallest clusters containing 1 node. Each cluster guarantees that every node in the cluster is not farther from its anchor than the coverage distance of 20.

Table 3-2: Distance matrix of the network in figure 3-2.

		Nodes												***TOTAL***
		A	*B*	*C*	*D*	*E*	*F*	*G*	*H*	*I*	*J*	*K*	*L*	
Facilities	*A*	0	15	37	55	24	60	18	33	48	40	58	67	455
	B	15	0	22	40	38	52	33	48	42	55	61	61	467
	C	37	22	0	18	16	30	41	28	20	58	39	39	348
	D	55	40	18	0	34	12	59	46	24	62	43	34	427
	E	24	38	16	34	0	36	25	12	24	47	37	43	336
	F	60	52	30	12	36	0	57	42	12	50	31	22	404
	G	18	33	41	59	25	57	0	15	45	22	40	61	416
	H	33	48	28	46	12	42	15	0	30	37	25	46	362
	I	48	42	20	24	24	12	45	30	0	38	19	19	321
	J	40	55	58	62	47	50	22	37	38	0	19	40	468
	K	58	61	39	43	37	31	40	25	19	19	0	21	393
	L	67	61	39	34	43	22	61	46	19	40	21	0	453

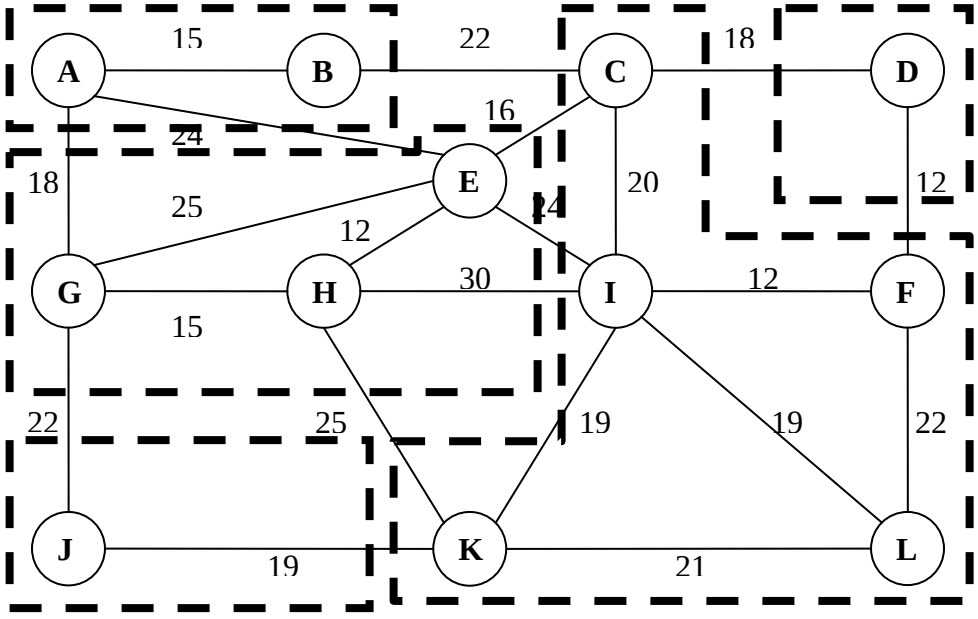

Figure 3-3: The clustering result of the network in figure 3-2 using set covering clustering method.

3.2.3.2 Greedy *d*-median algorithm

The greedy *d*-median algorithm is very similar to the greedy set covering algorithm outlined in the previous section. Again, we will illustrate the algorithm by using the example from the

network in Figure 3-2. The coverage distance is still 20. Starting from the distance matrix in Table 3-2, by summing the entries in each, we obtain the total connection cost for *1*-median problems. The smallest total connection cost is 321, when node *I* is selected as a facility. The average distance is 321/12 or 26.75. (Note that there are 12 nodes in the network.) The farthest distance is 48 from *A* to *I*, which still exceeds the coverage distance d_c of 20. The algorithm continues.

Table 3-3: Selecting the second median.

		Nodes												***TOTAL***
		A	*B*	*C*	*D*	*E*	*F*	*G*	*H*	*I*	*J*	*K*	*L*	
Facilities	*A*	0	15	20	24	24	12	18	30	0	38	19	19	219
	B	15	0	20	24	24	12	33	30	0	38	19	19	234
	C	37	22	0	18	16	12	41	28	0	38	19	19	250
	D	48	40	18	0	24	12	45	30	0	38	19	19	293
	E	24	38	16	24	0	12	25	12	0	38	19	19	227
	F	48	42	20	12	24	0	45	30	0	38	19	19	297
	G	18	33	20	24	24	12	0	15	0	22	19	19	206
	H	33	42	20	24	12	12	15	0	0	37	19	19	233
	I	48	42	20	24	24	12	45	30	0	38	19	19	321
	J	40	42	20	24	24	12	22	30	0	0	19	19	252
	K	48	42	20	24	24	12	40	25	0	19	0	19	273
	L	48	42	20	24	24	12	45	30	0	38	19	0	302

Once node *I* is selected, we locate the second median by updating each entry of the distance matrix by calculating $min\{d_{Ij}, d_{ij}\}$. In other words, node *j* connects to facility *I* if the connection cost is lower than connecting to the current facility *i*. Table 3-3 shows the resulting update. The second facility that we will select is node *G* since it yields the minimum connection cost of 206. Now, the farthest distance reduces to 33 from *B* to *G*, and the average distance reduces to 17.17. Since the farthest distance still exceeds the coverage distance, we need to add more facility to our solution. We again update the distance matrix by calculating $min\{d_{Gj}, d_{Ij}, d_{ij}\}$ for each node / candidate location pair *(i, j)*, and find that the third facility is located at node *C*, which yields the minimum connection cost of 161. Table 3-4 shows the update. Now the farthest distance is 22

and the average distance is 13.42. The algorithm proceeds in this manner until the distance from any node to the nearest facility are lower than or equal to 20.

Table 3-4: Selecting the third median.

		Nodes												***TOTAL***
		A	*B*	*C*	*D*	*E*	*F*	*G*	*H*	*I*	*J*	*K*	*L*	
Facilities	*A*	0	15	20	24	24	12	0	15	0	22	19	19	170
	B	15	0	20	24	24	12	0	15	0	22	19	19	170
	C	18	22	0	18	16	12	0	15	0	22	19	19	161
	D	18	33	18	0	24	12	0	15	0	22	19	19	180
	E	18	33	16	24	0	12	0	12	0	22	19	19	175
	F	18	33	20	12	24	0	0	15	0	22	19	19	182
	G	18	33	20	24	24	12	0	15	0	22	19	19	206
	H	18	33	20	24	12	12	0	0	0	22	19	19	179
	I	18	33	20	24	24	12	0	15	0	22	19	19	206
	J	18	33	20	24	24	12	0	15	0	0	19	19	184
	K	18	33	20	24	24	12	0	15	0	19	0	19	184
	L	18	33	20	24	24	12	0	15	0	22	19	0	187

In our example, the greedy *d*-median algorithm stops after five facilities were selected: *I*, *G*, *C*, *A* and *K*, respectively. The farthest distance is 19 and the average distance is 9.5. Figure 3-4 shows the clustering result of the greedy *d*-median algorithm.

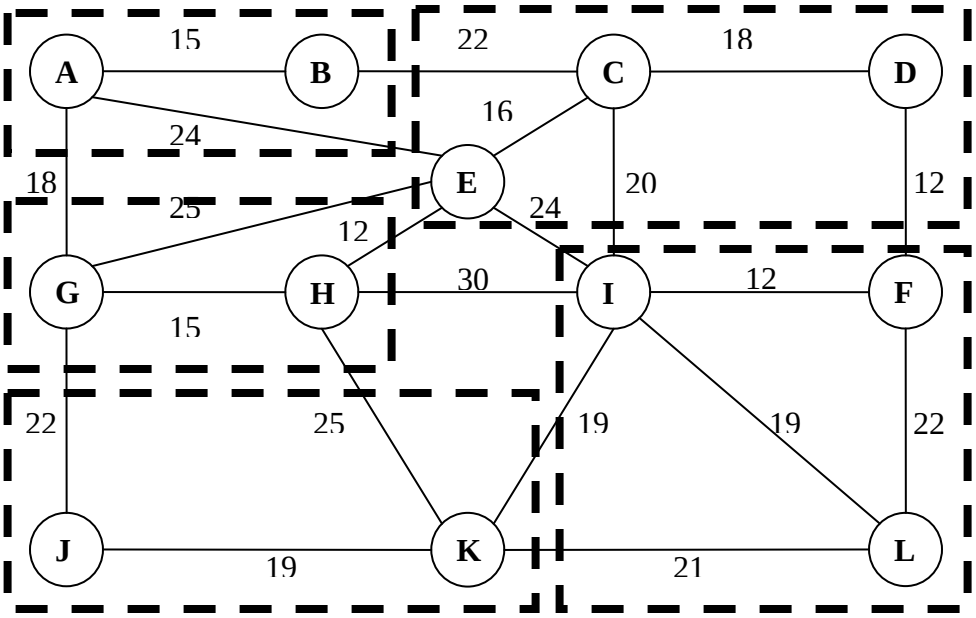

Figure 3-4: The clustering result of the network in figure 3-2 using d-median clustering method.

3.2.4 Performance metrics

In the study of the behavior of the two clustering methods, the performance metrics of interest are the average cluster size, the number of clusters, the number of effective clusters, the total connection cost, and the average radius of clusters. The following are the descriptions of each performance metrics. The average cluster size and the number of clusters are straightforward as the names suggest; the average number of nodes in the clusters and total number of resulting clusters, respectively. However, from our experience, using solely the total number of clusters may be illusive since there may exist many one-node clusters in power-law graph. Therefore, we introduce the number of effective clusters; i.e., the total number of clusters that contain more than one node. Both number of clusters and number of effective clusters are shown in the percentage of clusters to total number of nodes in the network. The total connection cost is calculated from summing the delays (shortest distances) from every node to its nearest anchor. The average radius of clusters is the average of the delay from anchor to the farthest node in the cluster. Table 3-5 summarizes the descriptions of these parameters.

Table 3-5: Summary of the performance parameters.

Performance parameters	*Descriptions*
Average cluster size	Average number of nodes in clusters
Number of clusters (%)	Total number of clusters
Number of effective clusters (%)	Total number of clusters consisting > 1 nodes
Total connection cost	The summation of delays in all clusters
Average radius of clusters	Average of the largest delay of every clusters

For example, the clustering result illustrates the Figure 3-4, the average cluster size is 2.4, total number of clusters is 5, total number of effective clusters is 5, total cost is 114, and the average radius of clusters is 17.2.

3.3 Results and analysis

The experiments were done as follows. For each edge delay distribution and each network size, we run the greedy set covering algorithm and greedy *d*-median algorithm using various delay bound input. Then, we observe the five performance parameters by plotting the results, shown in appendix A. The performance parameters are indicated in vertical axis of the graphs. A set of graphs consists of four sub-graphs; each corresponds to one of four different edge delay distributions on links of the underlying network. The four network sizes are also shown in each sub-graph. The delay bound input is shown in the normalized unit of *the mean of the edge delay distribution*. That is, the delay bound is displayed in the unit of the multiple of average delay of one hop.

Sensitivity analysis on the network size: It is easy to see from the resemblance of the plots that the changing in network size does not impact the behavior of the clustering methods. To confirm this fact, we find the correlation coefficients among the plots. The ranges of correlation coefficients are listed in the Table B-1 in appendix B. We can see that the correlation coefficients

are very high in every case, which support our hypothesis that the network size does not affect the behavior of clustering methods, given that the networks are power-law graphs. This suggests that no matter how large the Internet grows, our results are still valid.

Sensitivity analysis on edge delay distribution: On the other hand, the sensitivity analysis on edge delay distributions is more subtle to see. Again, we find the ranges of correlation coefficients and show in Table B-2. We observe that the correlation coefficients are not as high as those we found when we do the sensitivity analysis on network size. The correlation coefficients are notably low, as 0.3, when we consider the effective number of clusters, which implies that there is no correlation at all. However, the overall correlation is strong enough to distinguish the performance between set covering and *d*-median. The following are some observations we found. (1) No matter what edge delay distribution is used, set covering always yields the larger average cluster size. (2) The knee points of the plots of *d*-median always occur at the lower delay bound in the plot of the number of clusters. (3) Both set covering and *d*-median plots have the peaks in the plot of the number of effective clusters. However, *d*-median always has a larger number of effective clusters. (4) The set covering usually yields a larger total cost. *d*-median may have a higher total cost when the delay bound is smaller than 2 unit, but with very little amount. In every case, we can observe the knee points follows by the *stable range* (the change of the total cost is very small as the delay bound increases) in the plot of *d*-median on total cost. (5) The average radius of clusters obtaining from set covering is usually higher and more divergent.

These facts will be investigated further as we proceed. Even if the correlation coefficients cannot visibly ensure the similarity of the clustering results among different delay distribution, we can somehow capture the essential properties of the clustering methods that do not vary as the delay distribution changed. At this point, we conclude that the change in edge delay distribution does not affect much the behavior of the clustering method, especially, the *d*-median.

The practical delay bound (upper bound): Consider the plot of average cluster size, we can see that for both set covering and *d*-median, the average cluster size saturates at some point, i.e., when the cluster size equals to network size. At this point, the delay bound is so large that single cluster dominates the network. The evidence is also shown in the plot of number of clusters. At the point of saturate, the number of clusters decreases to one or two. In every case, the saturate

point falls around 4 units of delay bounds. Therefore, we suggest that the delay bound should not larger than 4 times of the average delay of one hop.

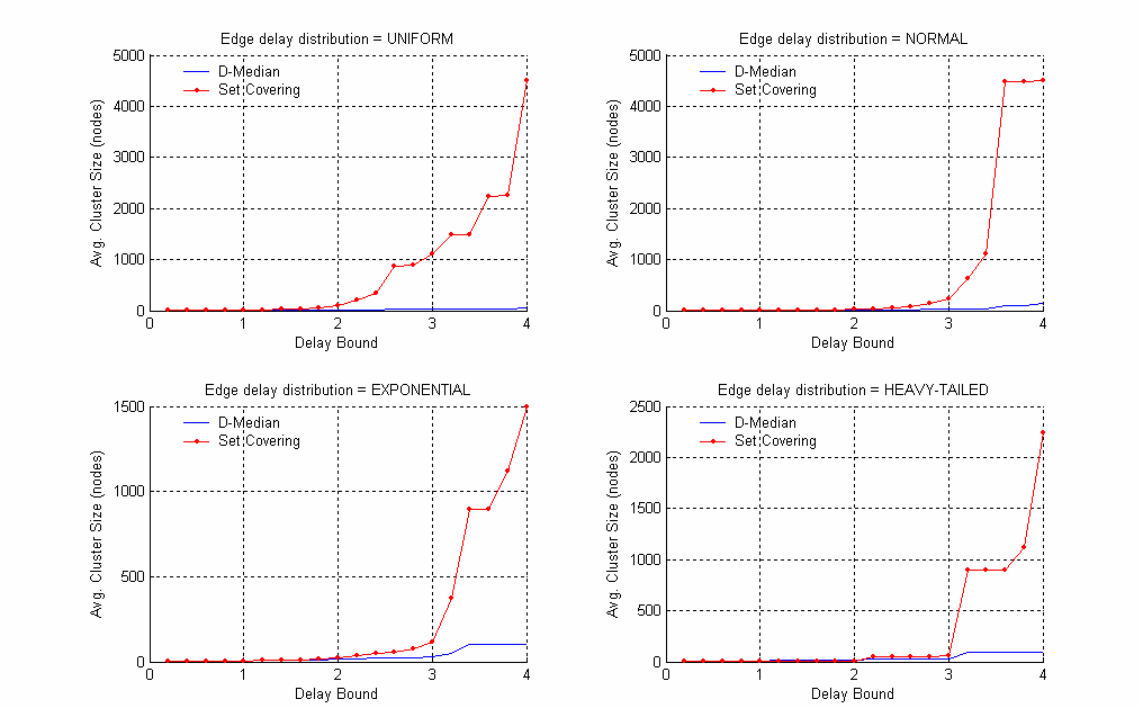

Figure 3-5: Average cluster sizes of 4,500-node networks

Number of clusters: Consider the plots of the number of clusters in Figure 3-6 (which shows only the network size of 4,500 nodes). The number of clusters decreases as the delay bound increases. This is due to the fact that the larger coverage-area clusters cover more nodes so that the number of total clusters is reduced. In every plot, we can see that the number of clusters rapidly decreases as delay bound increases in the beginning. At some point, the decreasing rate reduces. We call this point a *knee point*. The knee point is more obvious in the plot of *d*-median than in set covering. After the knee point, increasing the delay bound does not reduce much the number of clusters or we can say the number of clusters becomes stable.

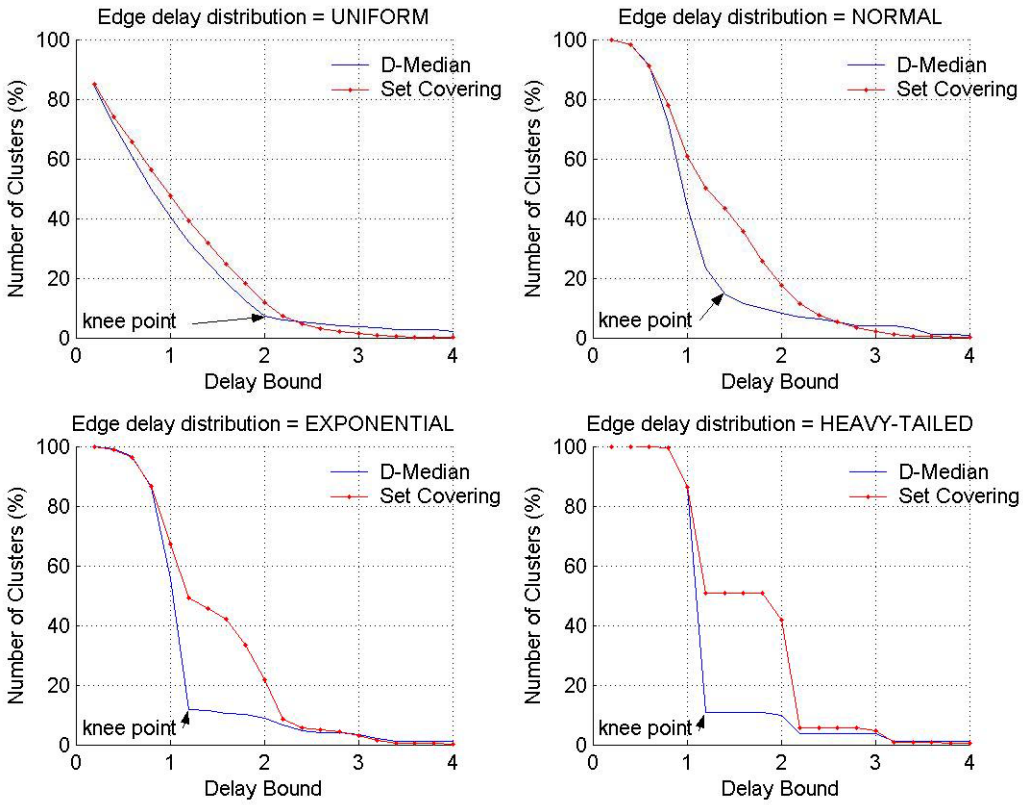

Figure 3-6: Number of clusters of 4,500-node networks

In general, we need the number of clusters to be as small as possible for scalability of the selective probing scheme. The number of clusters reflects the number of entries in the routing table and amount of the routing overheads. Also, we need the small delay bound to reduce the error from the estimation. Hence, the knee point seems to be an optimal point that balances both number of clusters and delay bound. We propose that the knee point should be treated as a lower bound of the delay bound input. From our observation on four different edge delay distribution, the knee point falls in the range from 1 to 2 times of the average delay of one hop. Therefore, for efficiency, the delay bound input should be, at least, larger than the average delay of the links in the network.

Note that the knee points of the plots of *d*-median are easier to see and always yield smaller number of clusters and smaller delay bound, at the same time, compared to those of set covering.

Actually, in most cases, the plots of *d*-median usually locate at the inner side, nearer to both axes than the plot of set covering. Consequently, we can say that *d*-median clustering method is more preferable than set covering since it usually yields smaller number of clusters for the given delay bound input.

Number of effective clusters: In power-law graph, large number of nodes is densely located while the rest are located sparsely. Sometimes, clustering the nodes in the remote area causes the resulting clusters containing only one node. The one-node clusters may not be efficient for our selective probing scheme. Many times, we find that increasing the delay bound a little bit, these one-node clusters can be merged with other clusters. However, if the one-node clusters are located very far from other nodes in the network, we should allow these one-node clusters to keep the error from the estimation small.

Now, we look at another performance parameter called number of effective clusters. The number of effective clusters gives the number of clusters that contains more than one node. From the plot in appendix A, we can see the peaks in both plots of *d*-median and set covering. Starting with a too small delay bound input, the number of effective clusters is very small since most of the clusters cover only one node. As the delay bound input increases, the number of effective clusters increases, while the total number of clusters decreases. This is because the larger bound covers more nodes. At some point, the number of effective clusters starts to decrease since the bound is larger and less number of clusters is required to cover the entire network. Note that the plots skew to the right, i.e., they are steeper when the delay bound is small. The peaks of the two methods fall in the range from 0.5 to 1.5 unit of delay bound. We suggest that the delay bound input should be higher than the peak points. At this point, the delay bound input seems to be large enough to eliminate all the unnecessary one-node clusters. However, we cannot say that the peaks are the optimal point for the delay bound input since every point on the plot shows the optimum for the corresponding delay bound input.

Furthermore, we can see that the number of effective clusters obtained from *d*-median is always higher than set covering's in every case. We also plot the ratio of the number of effective clusters over total number of clusters for the network size of 4500 nodes in Figure 3-7. The ratio of the *d*-median is apparently higher than set cover's. This, again, indicates that *d*-median clustering method is more efficient than set covering.

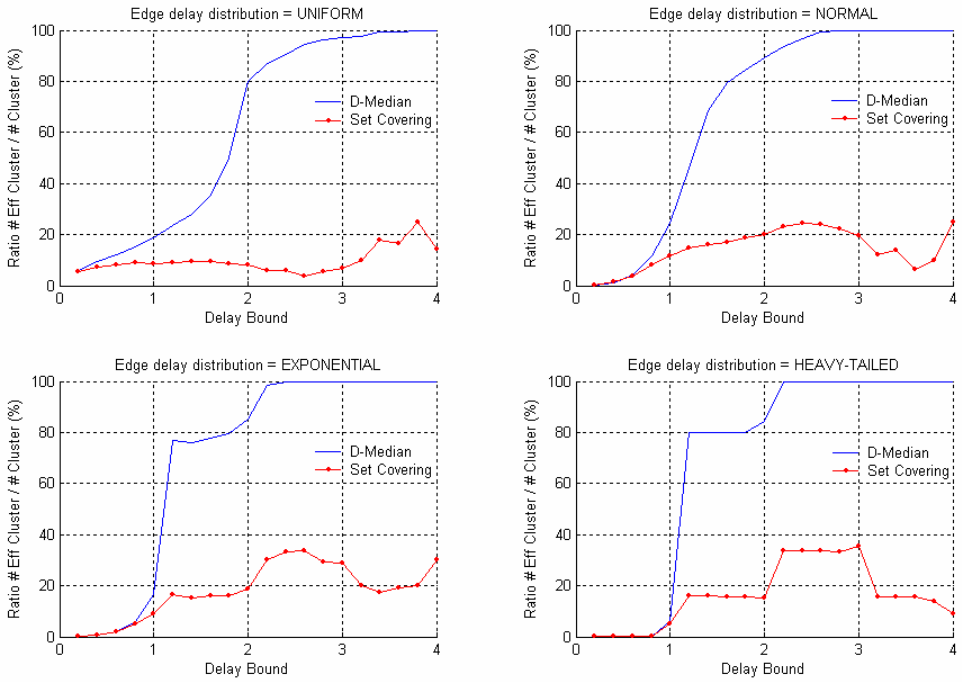

Figure 3-7: Ratio of the number of effective clusters per the total number of clusters

Total cost: Total cost indicates sum of the delay occur in the clustering result. In general, small total cost is desirable. The objective function of *d*-median tries to minimize this total cost. Therefore, given a same delay bound input, the total cost of *d*-median should be less than or equal to the total cost obtaining from set covering. Our observation from the plots in appendix A confirms this fact in most cases. However, in rare cases, we also see some points that *d*-median yields the larger total costs. This might cause from the error of the approximation algorithms.

The total cost divided by number of nodes in network yields average cost per node, i.e., the average delay between a node and its nearest anchor. Figure 3-8 shows the average cost for the network size of 4500 nodes. The unit of the average cost is the same unit as the delay bound. We can see the knee points in both plots of total cost and average cost. Note that the knee points here locate exactly the same point as in the plot of number of clusters. This implies that the knee point is the point where the clustering results become stable; the cost and the number of clusters are not much changed as the delay bound increases. The slope after the knee point is quite flat. We

suggest that the practical delay bound should be in the range starting from knee point to 4 units (4 times of the average delay of one hop).

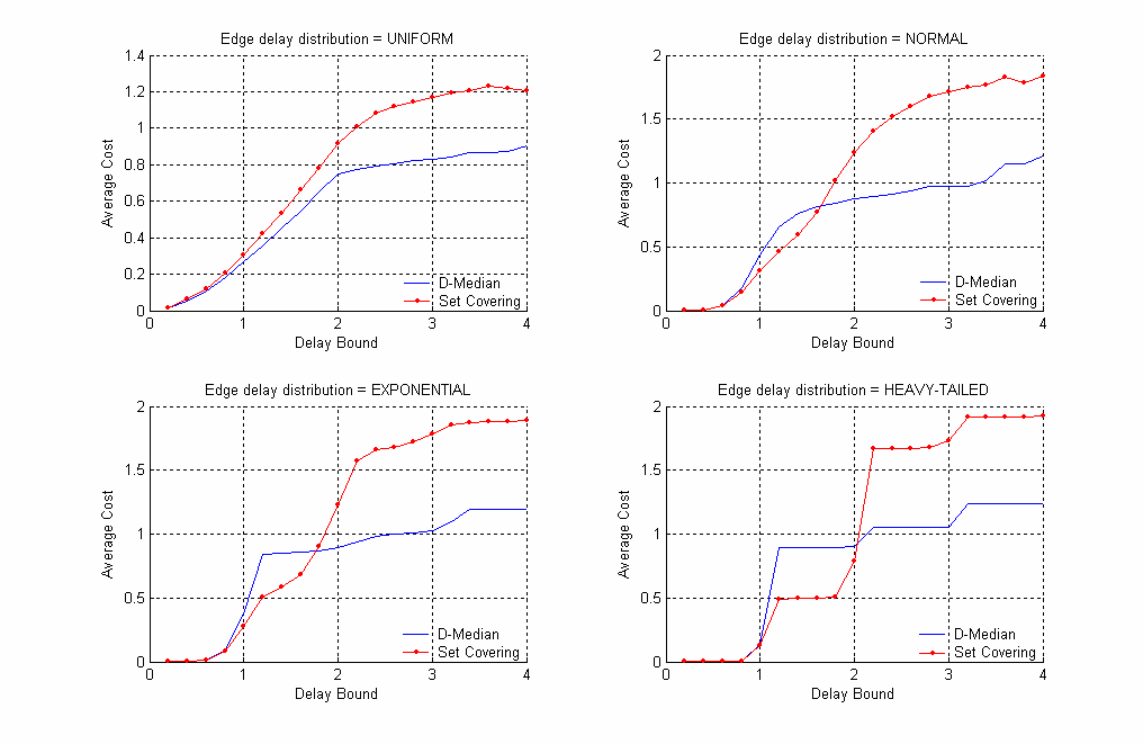

Figure 3-8: Average cost of 4,500-node networks

Consider again the average delay from a node to its nearest anchor in Figure 3-8. Given an appropriate delay bound input, *d*-median yields average delay around (and tending to be lower than) 1 hop. This reflects the loosely hierarchical nature of the Internet topology. However, average delay is around 2 hops in the case of set covering.

Radius: Although the average delay from a node to its nearest anchor is around 1-2 hops, the maximum delay, called a radius of a cluster, is more interesting since it indicates the worst case error that may occur from delay estimation of our selective probing scheme. The delay bound input implicitly defines the radius of the clusters. Ideally, the plot of the delay bound input and the radius of a cluster should be linear. Appendix A shows our results. The performance of *d*-median and set covering are quite indistinguishable in most cases. However, we find that the

Coefficient of Variation (CV) of the radius obtaining from *d*-median clustering method is smaller than those of set covering. The CV of the radius obtaining from *d*-median is in the range of 0.05-0.44, while the CV of the radius obtaining from set covering is in the range of 0.04-1.07. The smaller CV means that the clusters have about the same radius, which is useful for the network administrator or applications to approximate the error from estimation correctly. Hence, this is once again an evidence for supporting that *d*-median clustering method is preferable to set covering.

3.4 Conclusion

In this chapter, we study the behavior of set covering and *d*-median clustering methods over various Internet-like network instances, through many performance metrics. Nevertheless, the sensitivity analysis shows that networks sizes and delay distributions have minimum or *very small* effect to the clustering results. The study also found that *d*-median outperforms set covering in most cases. *d*-median clustering method usually yields smaller number of clusters but larger number of effective clusters than set covering. *d*-median usually yields the smaller total cost and average cost, compared to set covering. In addition, the cluster sizes obtaining from *d*-median have smaller deviation than those obtaining from set covering.

We also tried to define the range of practical delay bound input. First, we suggest that the upper bound should be around 4 times of the average delay per link in the network, where the network is likely to be dominated by a single cluster. The lower bound should be approximately around 1 to 2 times of the averaged delay per link. This is derived from the knees of the plots of number of clusters and the plots of total cost. Providing the delay bound input in this range, the clustering results do not change much as delay bound increases. This may be looked as an error tolerance property of the clustering method. Finally, we conclude that our proposed *d*-median clustering method is better suited for power-law graphs than any other clustering methods.

4.0 SELECTIVE PROBING FRAMEWORK

Last chapter, we partition the network into delay based classes of equivalence called clusters using one of the clustering methods: set covering or *d*-median. Both clustering methods require the complete knowledge of the underlying network graph in a form of distance matrix. In addition, they require maximum coverage distance or maximum delay bound for every cluster as an input. We have tried the two clustering methods with many Internet-like generated graphs, which are varied in sizes and delay distributions on links. The clustering results show that the effects from these factors are negligible. We also observed that practical delay bound input should be around 1 to 4 times of average delay per link in the network. Finally, we found that *d*-median clustering method often performs better than set covering and we suggested the use of *d*-median for our selective probing scheme.

Once the network is clustered, QoS metrics of any node can be retrieved in scalable way. This chapter describes the techniques of clustering-based QoS metrics estimation, focusing on delay metric. We also explain the selective probing framework which exploits these estimation techniques.

4.1 The clustering-based metrics estimation service

Once we applied one of the clustering methods to the network, each cluster is represented by an anchor node. These anchors play an important role in QoS metrics estimation. A considerably small number of anchors, about 9%-16% from the total number of nodes or less, makes it possible to actively and continuously probe for QoS metrics among themselves. The QoS measurements of the rest of the nodes in the network can be directly inferred from its anchor's.

In this work, we focus particularly on delay measurement apart from other measurements, e.g., number of hops, loss rate, delay jitter, bandwidth, policy flags, etc. Generally, the delay measurement mentioned here may be obtained from ICMP round-trip time (RTT) or TCP initial connection time.

We estimate delay between any pair of nodes *i* and *j*, associating with anchors a_i and a_j, respectively. If they belong to the same cluster, i.e., $a_i = a_j$, the estimated delay equals to the radius of the cluster. Otherwise, the estimated delay equals to measured delay between a_i and a_j. The former is called Intra-cluster estimation and the latter is called Inter-cluster estimation.

The simplest way is to use the last measured delay directly as addressed above. However, a transient conditions change in the network may cause the measured delay to be either too high or too low. And, by the time we use the measured delay information, it is usually no longer valid due to the highly dynamic nature of the Internet. Accordingly, for preciseness, we need to estimate the current round-trip delay based on their past measurements. One approach is by taking an average of the past *k* measured round-trip times. The simple averaging method can be expressed as follows:

$$ARTT(k+1) = \frac{1}{k+1}\sum_{i=1}^{k+1} RTT(i)$$

Where *RTT(i)* is the round-trip time observed for the *i*th observation, and *ARTT(k)* is the average round-trip time for the first *k* observations. This expression can be rewritten as follows:

$$ARTT(k+1) = \frac{k}{k+1} ARTT(k) + \frac{1}{k+1} RTT(k+1)$$

With this formulation, it is not necessary to recalculate the entire summation each time. Typically, however, we would like to give greater weight to more recent instances because they are more likely to reflect future behavior. A common approach, specified in [INFO81] for estimating the round-trip time for TCP, can be defined as follows:

$$SRTT(k+1) = \alpha \times SRTT(k) + (1-\alpha) \times RTT(k+1)$$

Where *SRTT(k)* is called the smoothed round-trip time estimate. α is a constant and $0 < \alpha < 1$. Generally, α is in the range of 0.8 to 0.9.

In contrast to arithmetic mean defined above, the usage of geometric mean is found to be practical in [CHEN02] where the delay distribution is assumed to be heavy-tailed. The geometric mean of k values is obtained by multiplying the values together and taking the kth root of the product. This can be expressed as follows:

$$GRTT = \left(\prod_{i=1}^{k} RTT(i) \right)^{1/k}$$

The engineering problems, such as how often should an anchor probe for the metrics from other anchors, or how large the window size k should be in above formulas, are yet to be investigated. Further study on the supporting protocol must clarify these problems. Nevertheless, we suggest that the clustering task should be done by a dedicated server and distributing the results to the anchors. The clustering calculation, from our experience, is a resource consuming task. Therefore, we also suggest that the calculation and results distribution should be done periodically, in the time scale of days or weeks rather than minutes or hours.

4.2 Selective probing QoS routing framework

From the clustering result and the knowledge of the underlying topology, we can build a virtual topology, where nodes represent by clusters and links represent the physical connections between clusters. We call this a meta-topology or meta-graph. Essentially, a meta-graph is a network of anchors since each anchor represents a cluster. Figure 4-1 shows a meta-graph deriving from the network in Figure 2-1 (b). Note that, this introduces the concept of hierarchical topology.

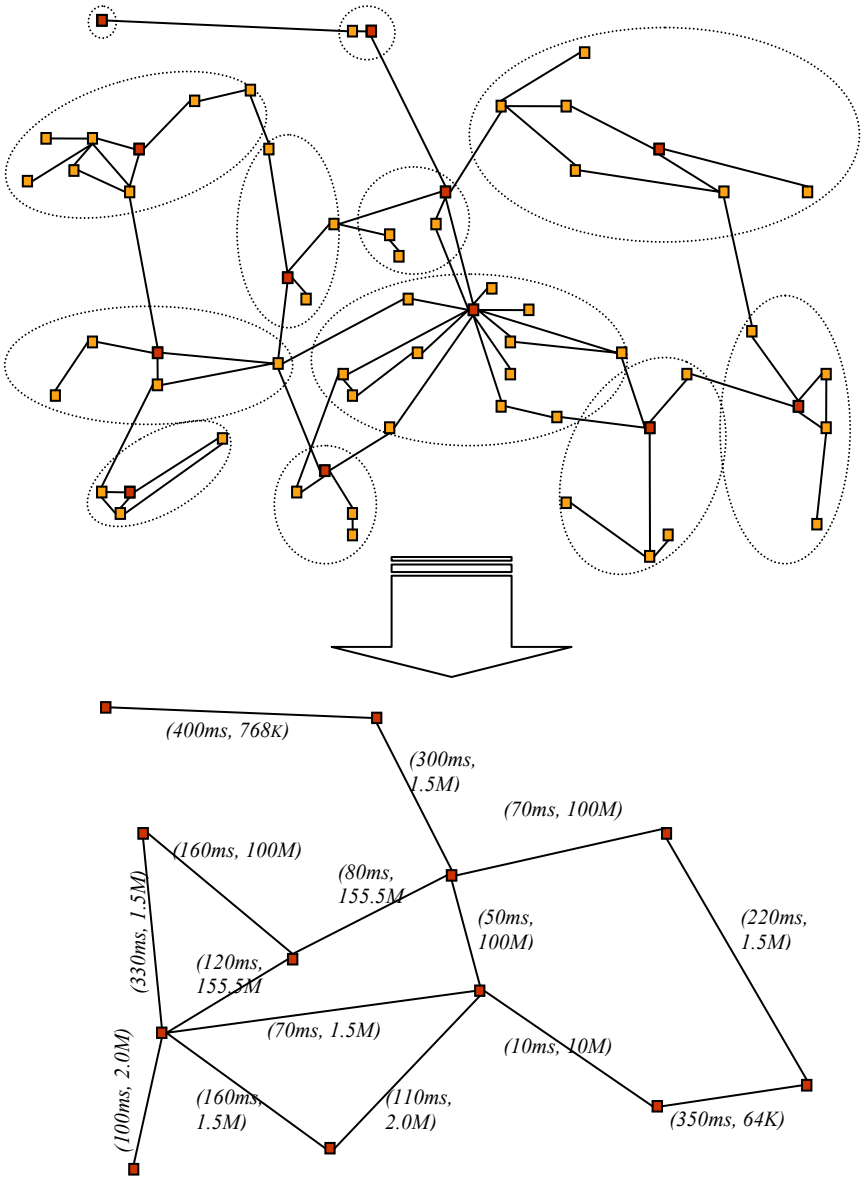

Figure 4-1: Deriving a meta-graph from the physical topology.

Once the network is clustered and meta-graph is inferred, selective probing is then performed to obtain the metrics information of the graph. The metrics estimation service described in the previous section can be used to maintain this information. Simple single-constrained routing algorithms, such as Dijkstra's or Bellman-Ford, or even the QoS routing algorithms that take into account multiple constraints can efficiently run on this higher hierarchy topology. The result may consist of single or multiple feasible paths. The advantage of this scheme is that there is no resource reservation mechanism required, which makes selective probing framework stateless and scalable. The framework can be properly used to enhance and extend DiffServ ability to support QoS in suggesting paths.

The objective of QoS routing is to find the feasible paths that conform to the multiple constraints requested by applications, i.e., the multi-constrained problem (MCP). One way to do so is by running multiple topology filtering procedures, each time with different metric. The topology filtering procedure propagates itself through the meta-graph, filtering out the paths that do not meet the given constraints. Essentially, the procedure for finding the feasible paths on the meta-graph can be summarized as follows:

Meta-graph path selection procedure

- Cluster the network using *d*-median clustering method to build a meta-graph.
- Selectively probe the anchors and estimate delay and other metrics using the averaging techniques discussed in previous chapter.
- Perform topology filtering procedure to find the feasible paths.
- Perform the shortest paths on the filtered topology to find the shortest path based on selected metric.

To further illustrate the basic operations of the procedure above, we consider again the meta-graph depicted in Figure 4-1. Through the example, we will detail the topology filtering procedure. We assume that the source is denoted by ***S*** and destination is denoted by ***D***, as shown in Figure 4-2. The label, *(Delay, Bandwidth)*, on the links denotes the estimated *delay* and estimated *bandwidth*, respectively. Suppose an application requests a path with the following three constraints:

1. hop-counts must be less than 20 hops,
2. bandwidth must be larger than or equal to 1 Mbps, and
3. total delay must be less than 500 ms.

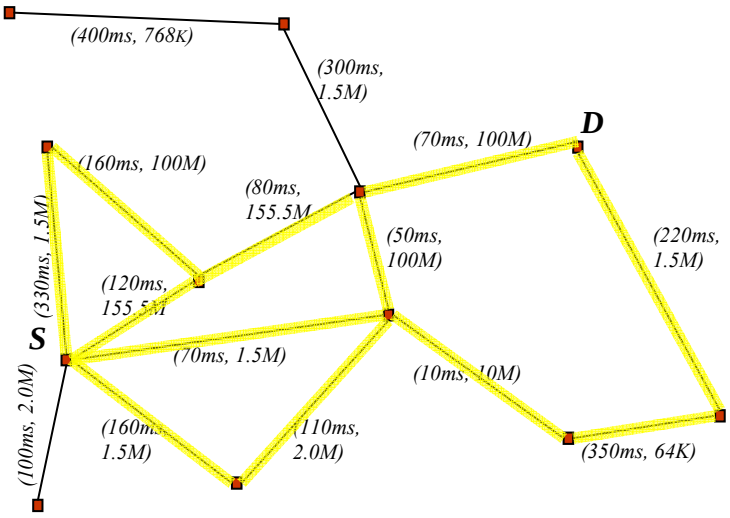

Figure 4-2: Topology filtering – number of hops.

By running the topology filtering procedure with the first constraint, we can filter some routes that do not lead to the destination within the given hop-counts. The feasible paths are highlighted in Figure 4-2. We then run the topology filtering procedure for the second time with the second constraint. This time, we filter out the paths that contain a 64Kbps link, located at the bottom right of the meta-graph. The feasible paths are shown in Figure 4-3. Finally, we run the topology filtering for the last time with the third constraint. The paths that the sum of delays exceeds 500 ms are eliminated from the set of feasible paths. At this point, we obtain three feasible paths denoted by arrows in Figure 4-4. Any of these paths can be used as a solution.

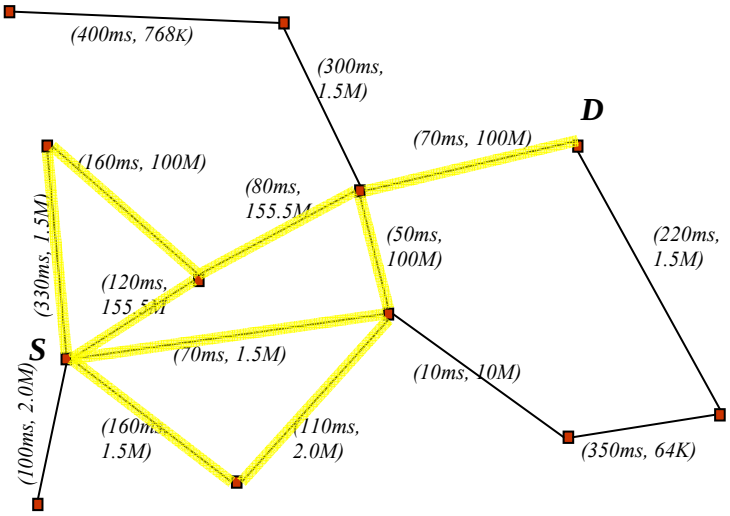

Figure 4-3: Topology filtering – bandwidth.

However, if we want the optimal solution selecting from these feasible solution, the problem is changed to a multi-constrained optimal path (MCOP) problem and we need to specify the principal constraint that describes the word optimal. For example, see Figure 4-4, if we want the optimal path in the sense of end-to-end delay, we just simply run the shortest path on the filtered meta-graph. The optimal solution will be the path denoted by the arrow in the middle. On the other hand, if the path with largest bandwidth is desired, the uppermost arrow path will be the optimal solution.

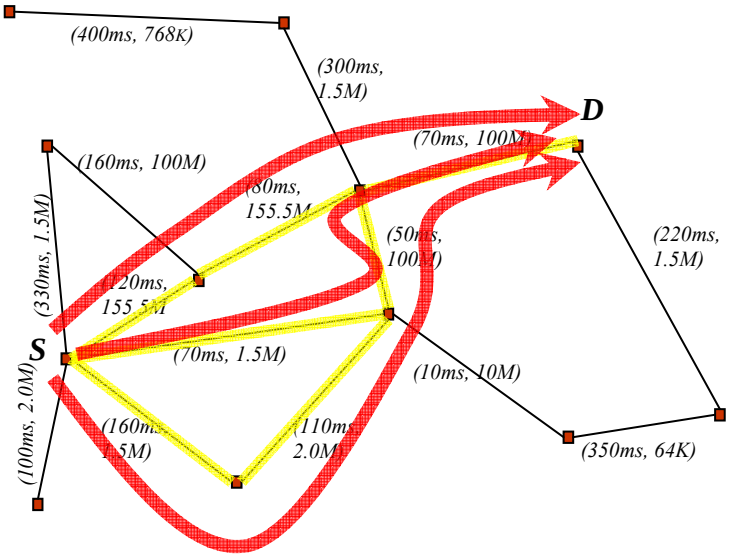

Figure 4-4: Topology filtering – delay.

The solution on the meta-graph shows the feasible paths that help suggesting the routing decision on the physical topology. Routers or nodes can exploit this information by forwarding the corresponding packets according to the suggested direction. Whereby, the physical route within the cluster is decided by local routing algorithm. The feasible paths are treated as preferred paths, which router may or may not follow. The solution path on a meta-graph can be used for overlay routing.

The QoS routing scheme described above is given as an example. Further details on the protocol implementation and the use of the solution are yet to be studied. One important note is that, at least, all the anchors must support our selective probing scheme, in order to corporate with each other.

4.3 Conclusion

In this chapter, we review the techniques used for the QoS metrics estimation of the clustered network. Specifically, we focus on the delay metric measurement and discuss some well-known

averaging techniques for delay. Then we give the concept of meta-graph used in our selective probing scheme and provide the example for the path selection calculation, using topology filtering. The QoS routing done on the meta-graph is used for suggesting the packet forwarding direction, which aids DiffServ ability to support QoS.

5.0 CONTRIBUTIONS AND FUTURE WORKS

5.1 Summary of contributions

In this work, we tried to develop a routing architecture that supports QoS in a more scalable way. For this, we proposed a clustering-based selective probing framework that partitions the nodes in the network into delay-based classes of equivalence, and abstracts the physical topology into a form of meta-graph. Scalability is the major concern of this work. Selective probing is done in a favor of balancing between routing overheads and routing information accuracy. The contributions of this work are summarized as follows:

1. It proposed the selective probing routing architecture which aimed to support QoS routing in a scalable way.
2. It reviewed four mathematical models of the clustering methods and their corresponding approximation algorithms.
3. It proposed *d*-median clustering method and its approximation algorithm.
4. It evaluated the behaviors of set covering and *d*-median clustering methods over the Internet-like power-law generated graphs.
5. It observed that the sizes of networks and the delay distributions on the graphs have none or least effect to both clustering methods. This validates the use of the clustering methods on the real Internet topology.
6. It observed that *d*-median outperforms set covering clustering method, at least from our set of performance metrics.
7. It suggested the practical input range for *d*-median clustering method.

5.2 Future works

Numerous open issues regarding the selective probing QoS routing scheme are yet to be investigated. Our work can be improved in several ways, including the following issues:

- As we have seen that both set covering and *d*-median clustering method require $n \times n$ distance matrix as an input, we need the method to reduce the input size so that the algorithm can run more efficiently for the large scale networks.
- The approximation algorithm for *d*-median clustering method may be investigated in more details and improved so that it yields the better approximation factor. The approximation algorithm proposed here has no guaranteed approximation factor. The algorithm is needed to be improved to be more precise while minimizes the time complexity.
- In this work, we gave the concept of the selective probing routing architecture. We focused only on the methods of clustering the network. However, the details of the entire protocol are yet to be defined. For example, the types of metrics collected, the path selection methods, and frequency of probing, are needed to be specified.

Based on our work, the future works can be diverse into many areas, including:

- Our QoS routing framework is done on the basis of unicast. To support the multicast multimedia applications, future work may try to develop the multicast QoS routing version, which aims to find the algorithm that establishes the QoS based multicast tree.
- The clustering methods discussed and developed in this work may be applied, as a fundamental component, in the area of sensor network. For sensor network, the strategically aggregated clustering may be used for the purposes of information aggregation or power reduction. Method of aggregation is still the active area of research.

- Future work can investigate into the area of security. It may address the issue for the case of inter-domain routing security management. For example, when the probing signal is crossing the administrative domains (ASs), there should be some mechanisms to validate both the probing and QoS routing information in terms of confidentiality, integrity, and authentication.

APPENDIX A

EXPERIMENTAL RESULTS

The following shows the observations of the five performance parameters: *Average cluster size*, *Number of clusters*, *Number of effective clusters*, *Total cost*, and *Average radius* and are listed on the vertical axes. Each performance parameter consists of four sub-plots. Each sub-plot denotes one of the edge delay distributions of the underlying graph. The edge delay distributions used in this work are *Uniformed*, *Normal*, *Exponential*, and *Heavy-tailed*. Furthermore, each sub-plot shows the performance of the *set covering* and *d-median* clustering methods in four different network sizes, which are 3,037 nodes, 3,500 nodes, 4,500 nodes, and 4,500 nodes.

The two clustering methods take the delay bound as an input. We intended to generalize the unit of the delay into the normalized form. The delay bound inputs are shown in the unit of the *multiple of the average delay per one hop*. That is, we find the average delay of the links in the network and use it as a unit of delay bound input. The observations are done by varying the delay bound inputs from very small to very large. Then, we observe simultaneously the clustering results, through the five performance parameters, of the two clustering methods.

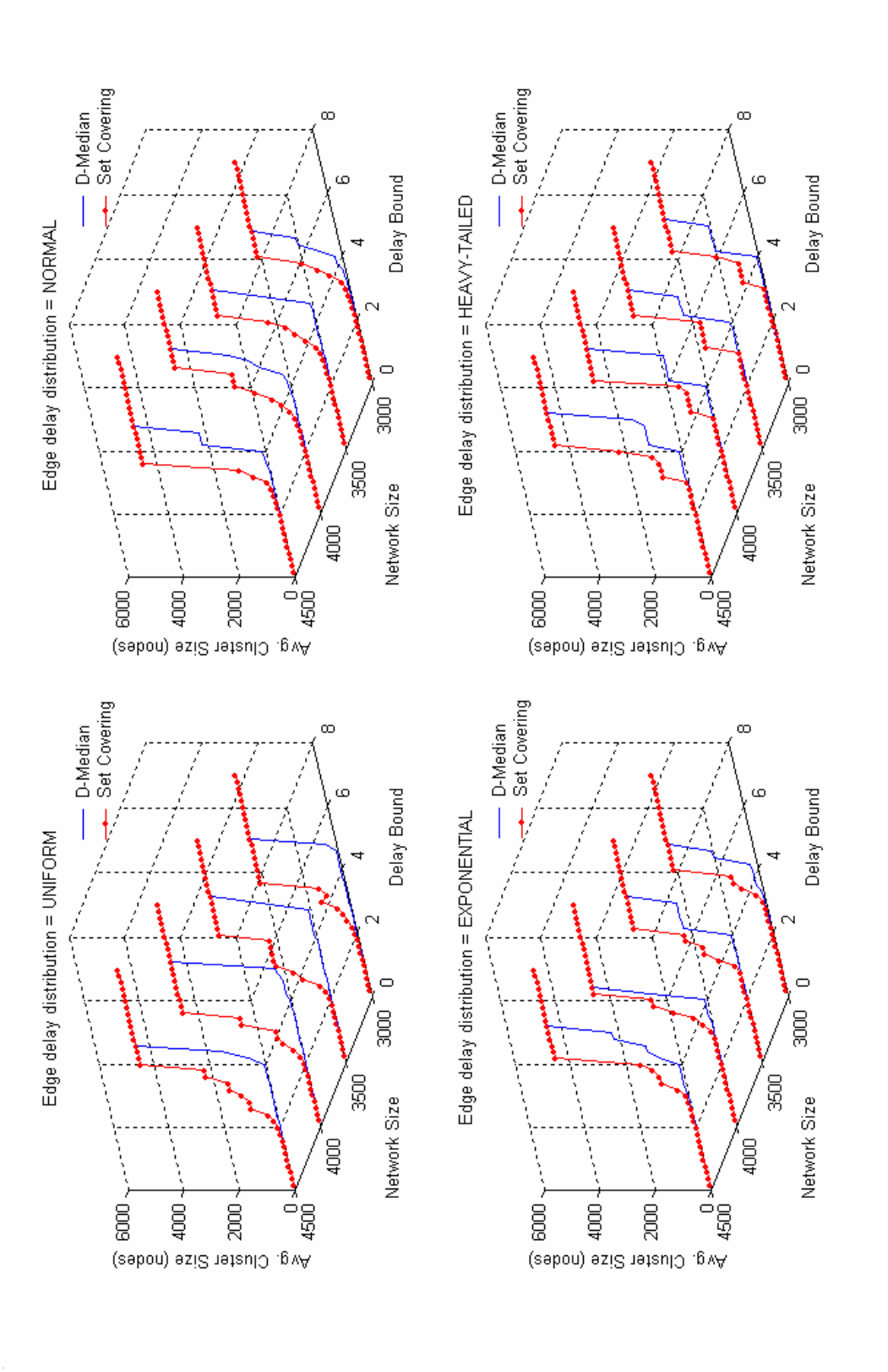

Figure A-1: Clustering results on average cluster size (nodes)

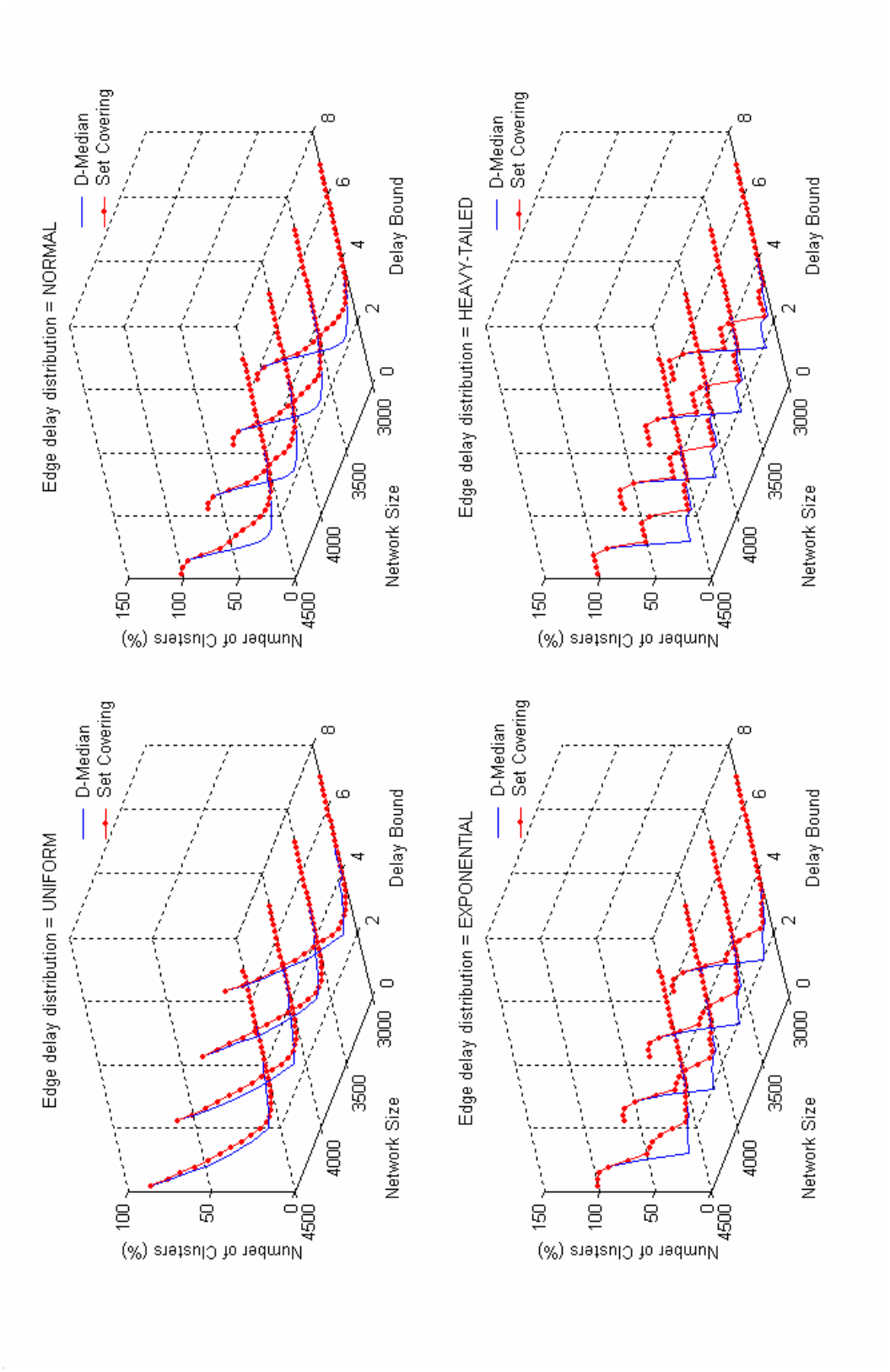

Figure A-2: Clustering results on number of clusters (%)

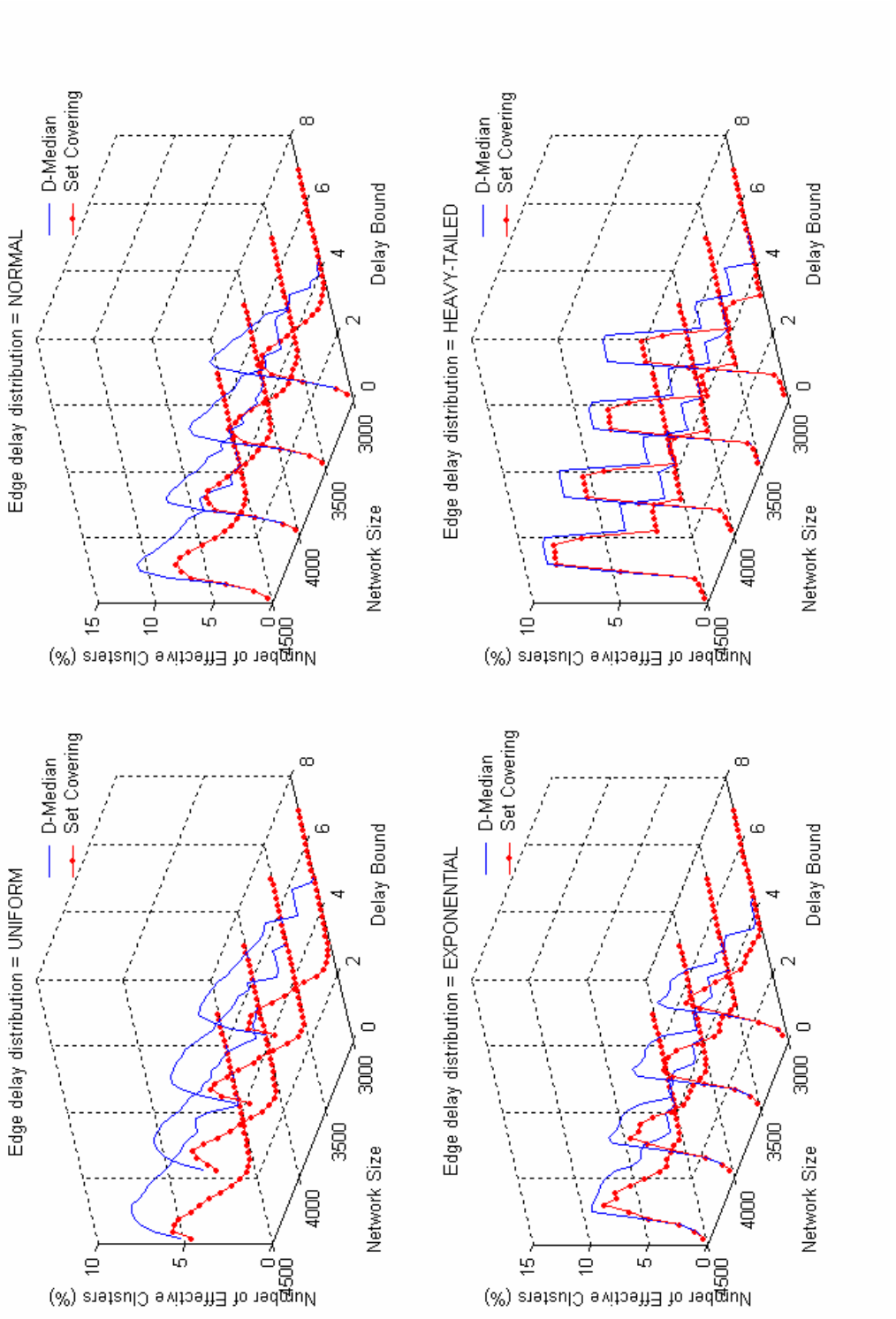

Figure A-3: Clustering results on number of effective clusters (%)

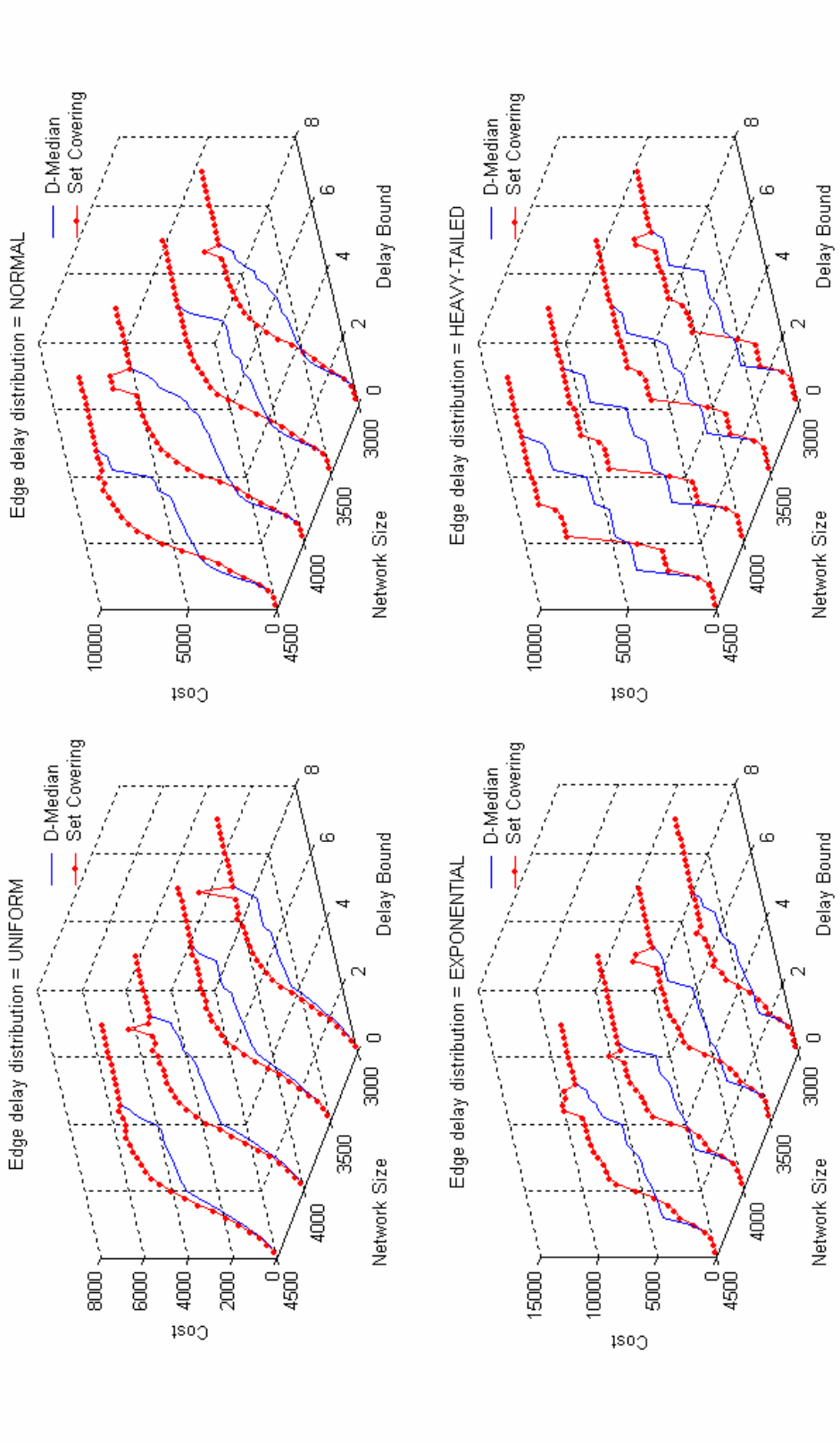

Figure A-4: Clustering results on connection cost

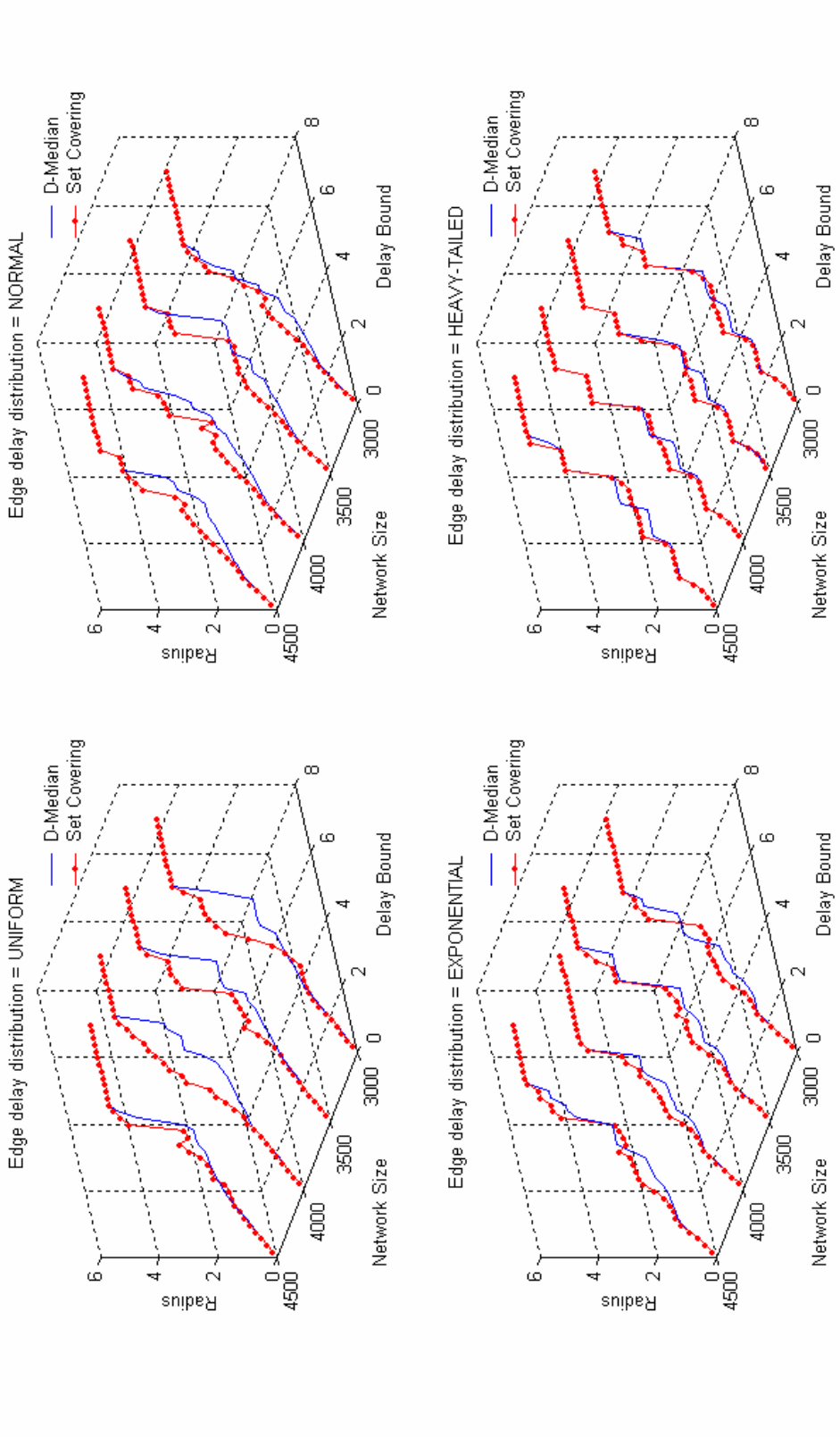

Figure A-5: Clustering results on average radius

APPENDIX B

SENSITIVITY ANALYSES

We perform the sensitivity analysis on network sizes and edge delay distributions by finding the coefficients of correlation among the plots. The coefficients of correlation are shown in the range of minimum and maximum, which give us the idea how similar the plots are.

Table B-1: Sensitivity analysis on *network sizes*

Performance metrics	*delay distributions*	*Clustering methods*	***Ranges of correlation coefficients***
Average cluster size	Uniformed	Set covering	0.9560-0.9930
		D-median	0.8376-0.9877
	Normal	Set covering	0.9014-0.9831
		D-median	0.9297-0.9954
	Exponential	Set covering	0.9719-0.9976
		D-median	0.9026-0.9837
	Heavy-tailed	Set covering	0.9946-0.9996
		D-median	0.9555-0.9981
Number of clusters	Uniformed	Set covering	0.9991-0.9999
		D-median	0.9996-0.9999
	Normal	Set covering	0.9995-0.9999
		D-median	0.9996-0.9999
	Exponential	Set covering	0.9996-0.9999
		D-median	0.9998-0.9999
	Heavy-tailed	Set covering	0.9997-1.0000
		D-median	0.9999-1.0000
Number of effective clusters	Uniformed	Set covering	0.9929-0.9976
		D-median	0.9864-0.9976
	Normal	Set covering	0.9948-0.9985
		D-median	0.9892-0.9939
	Exponential	Set covering	0.9949-0.9983
		D-median	0.9892-0.9965
	Heavy-tailed	Set covering	0.9970-0.9996
		D-median	0.9972-0.9998
Total connection cost	Uniformed	Set covering	0.9768-0.9993
		D-median	0.9847-0.9975
	Normal	Set covering	0.9902-0.9994
		D-median	0.9779-0.9957
	Exponential	Set covering	0.9847-0.9938
		D-median	0.9857-0.9963
	Heavy-tailed	Set covering	0.9941-1.0000
		D-median	0.9954-0.9995
Average radius of clusters	Uniformed	Set covering	0.9748-0.9904
		D-median	0.9324-0.9915
	Normal	Set covering	0.9645-0.9944
		D-median	0.9494-0.9934
	Exponential	Set covering	0.9800-0.9987
		D-median	0.9696-0.9939
	Heavy-tailed	Set covering	0.9919-0.9976
		D-median	0.9939-0.9996

Table B-2: Sensitivity analysis on *delay distributions*

Performance metrics	*Network size*	*Clustering methods*	***Ranges of correlation coefficients***
Average cluster size	3037	Set covering	0.9153-0.9981
		D-median	0.9223-0.9978
	3500	Set covering	0.9483-0.9954
		D-median	0.9314-0.9827
	4000	Set covering	0.8855-0.9888
		D-median	0.7872-0.9856
	4500	Set covering	0.9078-0.9975
		D-median	0.9020-0.9929
Number of clusters	3037	Set covering	0.9725-0.9977
		D-median	0.9445-0.9935
	3500	Set covering	0.9730-0.9970
		D-median	0.9418-0.9932
	4000	Set covering	0.9764-0.9973
		D-median	0.9426-0.9932
	4500	Set covering	0.9690-0.9975
		D-median	0.9428-0.9932
Number of effective clusters	3037	Set covering	0.3786-0.9770
		D-median	0.6962-0.9858
	3500	Set covering	0.3935-0.9783
		D-median	0.7124-0.9752
	4000	Set covering	0.3813-0.9788
		D-median	0.7064-0.9288
	4500	Set covering	0.3382-0.9757
		D-median	0.6916-0.9799
Total connection cost	3037	Set covering	0.9692-0.9941
		D-median	0.9640-0.9942
	3500	Set covering	0.8750-0.9978
		D-median	0.9616-0.9956
	4000	Set covering	0.9728-0.9923
		D-median	0.9569-0.9888
	4500	Set covering	0.9714-0.9973
		D-median	0.9748-0.9958
Average radius of clusters	3037	Set covering	0.9590-0.9915
		D-median	0.9387-0.9932
	3500	Set covering	0.9858-0.9977
		D-median	0.9603-0.9925
	4000	Set covering	0.9627-0.9843
		D-median	0.9827-0.9968
	4500	Set covering	0.9592-0.9955
		D-median	0.9827-0.9968

BIBLIOGRAPHY

[CAIDA] http://www.caida.org/tools/measurement/skitter/as_adjacencies.xml

[NS2] The Network Simulator. http://www.isi.edu/nsnam/ns/ns-documentation.html

[ORLIB] http://www.brunel.ac.uk/depts/ma/research/jeb/info.html

[ALON03] N. Alon, B. Awerbuch, Y. Azar, N Buchbinder, and J. Naor. The online set cover problem. In *Proceedings of ACM symposium on Theory of Computing*, June 9-11, 2003.

[APOS99a] G. Apostolopoulos, R. Guerin, and S. Kamat. Implementation and performance measurements of QoS routing extensions to OSPF. In *Proceedings of IEEE INFOCOM*, 21-25 March 1999.

[APOS99b] G. Apostolopoulos, R. Guerin, S. Kamat, A. Orda, T. Przygienda, and D. Williams. QoS Routing Mechanisms and OSPF Extensions. *RFC 2676*, Experimental RFC, Internet Engineering Task Force, August 1999.

[ARYA01] V. Arya, N. Garg, R. Khandekar, K. Munagala and V. Pandit. Local search heuristic for k-median and facility location problems. In *Proceedings of the thirty-third annual ACM symposium on Theory of computing*. July 2001.

[ATM99] ATM Forum Technical Committee. *Traffic Management Specification*, Version 4.1. AF-TM-0121.000, March 1999.

[ATM02] ATM Forum Technical Committee. *Private Network-Network Interface Specification Version 1.1 (PNNI 1.1)*. af-pnni-0055.001, April 2002.

[BARA99] A.-L. Barabasi and R. Albert. Emergence of Scaling in Random Networks. *Science*, pp. 509-512, Oct. 1999.

[BLAK98] S. Blake, D. Black, M. Carlson, E. Davies, Z. Wang and W. Weiss. An Architecture for Differentiated Services. *RFC 2475*, Internet Engineering Task Force, December 1998.

[BRAD94] R. Braden, D. Clark and S. Shenker. Integrated Services in the Internet Architecture: an Overview. *RFC 1633*, Internet Engineering Task Force, June 1994.

[BU02] T. Bu and D. Towsley. On distinguishing between Internet power law topology generators. In *Proceedings of IEEE INFOCOM*, 23-27 June 2002.

[CALV97] K. Calvert, M. Doar and E. W. Zegura. Modeling Internet Topology. *IEEE Communications Magazine*, June 1997.

[CHAR99a] M. Charikar, S. Guha, E. Tardos, and D. B. Shmoys. A constant-factor approximation algorithm for the *k*-median problem. In *Proceedings of the thirty-first annual ACM symposium on Theory of computing*, May 1999.

[CHAR99b] M. Charikar and S. Guha. Improved Combinatorial Algorithms for the Facility Location and k-Median Problems. In *Proceedings of the 40th Annual Symposium on Foundations of Computer Science*, October 1999.

[CHAR01] M. Charikar, S. Khuller, D. M. Mount and G. Narasimhan. Algorithms for facility location problems with outliers. In *Proceedings of the twelfth annual ACM-SIAM symposium on Discrete algorithms*. January 2001.

[CHEN98a] S. Chen and K. Nahrstedt. Distributed quality-of-service routing in high-speed networks based on selective probing. In *Proceedings of Local Computer Networks*, 11-14 Oct. 1998.

[CHEN98b] S. Chen and K. Nahrstedt. An overview of quality of service routing for next-generation high-speed networks: problems and solutions. *IEEE Network*, Volume: 12 , Issue: 6, Pages:64 - 79, November/December 1998.

[CHEN98c] S. Chen and K. Nahrstedt. Distributed QoS routing with imprecise state information. In *Proceedings of the 7th International Conference on Computer Communications and Networks*, Pages:614 – 621, 12-15 Oct. 1998.

[CHEN02] Y. Chen, K. H. Lim, R. H. Katz and C. Overton. On the stability of network distance estimation. *ACM SIGMETRICS Performance Evaluation Review,* v.30 n.2, September 2002.

[CRAW98] E. Crawley, R. Nair, B. Rajagopalan, and H. Sandick. A Framework for QoS-based Routing in the Internet. *RFC 2386*, Internet Engineering Task Force, August 1998.

[DASK95] M. S. Daskin. *Network and Discrete Location Model, Algorithms, and Applications*, John Wiley & Sons, Inc., 1995.

[DAVI02] B. Davie, A. Charny, J.C.R. Bennett, K. Benson, J.Y. Le Boudec, W. Courtney, S. Davari, V. Firoiu and D. Stiliadis. An Expedited Forwarding PHB (Per-Hop Behavior). *RFC 3246*, Internet Engineering Task Force, March 2002.

[FALO99] M. Faloutsos, P. Faloutsos, and C. Faloutsos. On power-law relationships of the Internet topology. *ACM SIGCOMM Computer Communication Review*, Volume 29, Issue 4, August 1999.

[FEIG98] U. Feige. A threshold of ln n for approximating set cover. *Journal of the ACM*, Vol. 45, No. 4, pp. 634-652, July 1998.

[GARE79] M. R. Garey and D. S. Johnson. *Computer and Intractability: A Guide to NP-Completeness*, W.H. Freeman, 1979.

[GOLD71] A. J. Goldman. Optimal Center Location in Simple Networks. *Transportation Science*, 5, 212-221, 1971.

[GROS97] T. Grossman and A. Wool. Computational experience with approximation algorithms for the set covering problem. *European Journal of Operational Research*. 101, pp. 81-92, 1997.

[GUER97] R. A. Guerin, A. Orda, and D. Williams. QoS routing mechanisms and OSPF extensions. In *Proceedings IEEE Global Telecommunications Conference* (GLOBECOM), Volume: 3, Pages:1903 - 1908, 3-8 Nov. 1997.

[GUER99] R. A. Guerin and A. Orda. Routing in networks with inaccurate information: theory and algorithms. *IEEE/ACM Transactions on Networking*, Volume: 7, Issue: 3, Pages:350 – 364, June 1999.

[GUHA00] S. Guha, A. Meyerson, and K. Munagala. Hierarchical placement and network design problems. In *Proceedings of 41st Annual Symposium on Foundations of Computer Science*, Pages:603 – 612, 12-14 Nov. 2000.

[GUHA98] S. Guha and S. Khuller. Greedy strikes back: improved facility location algorithms. In *Proceedings of the ninth annual ACM-SIAM symposium on Discrete algorithms*, January 1998.

[HAKI65] S. L. Hakimi. Optimum Distribution of Switching Centers in a Communication Network and Some Related Graph Theoretic Problems. *Operations Research*, 13, 462-475, 1965.

[HEIN99] J. Heinanen, F. Baker, W. Weiss and J. Wroclawski. Assured Forwarding PHB Group. *RFC 2597*, Internet Engineering Task Force, June 1999.

[HOCH82] D. S. Hochbaum, *Heuristics for the fixed cost median problem. Mathematical Programming*, 22 (2): 148-162, 1982.

[INFO81] Information Sciences Institute. Transmission Control Protocol. *RFC 793*, Internet Engineering Task Force, September 1981.

[JAFF84] J. Jaffe, Algorithms for finding paths with multiple constraints. *Networks*, vol. 14 pp. 95-116, 1984.

[JAIN99] K. Jain and V. V. Vazirani. Primal-dual approximation algorithms for metric facility location and *k*-median problems. In Pro*ceedings of 40th Annual IEEE Symposium on Foundations of Computer Science*, Pages:2 - 13, 17-19 Oct. 1999.

[JAIN00] K. Jain and V. V. Vazirani. An approximation algorithm for the fault tolerant metric facility location problem. In *Proceedings of the Third International Workshop on Approximation Algorithms for Combinatorial Optimization*, September 2000.

[JAIN02] K. Jain, M. Mahdian and A. Saberi. A new greedy approach for facility location problems. In *Proceedings of the thiry-fourth annual ACM symposium on Theory of computing*, May 2002.

[JAIN03] K. Jain, M. Mahdian, E. Markakis, A. Saberi and V. V. Vazirani. Greedy facility location algorithms analyzed using dual fitting with factor-revealing LP. *Journal of the ACM (JACM)*, Volume 50, Issue 6, November 2003.

[JAMI00] S. Jamin, C. Jin, Y. Jin, D. Raz, Y. Shavitt, and L. Zhang. On the placement of Internet instrumentation. In *Proceedings of IEEE INFOCOM,* 2000, 26-30 March 2000.

[JOHN74] D. Johnson. Approximation algorithms for combinatorial problems. *Journal of Computer System Science*, 9, pp. 256-278, 1974.

[KARI79] O. Kariv and S. L. Hakimi. An Algorithmic Approach to Network Location Problems. II: The p-Medians. *SIAM Journal on Applied Mathematics*, 37, 539-560, 1979.

[KORK03a] T. Korkmaz and M. Krunz. Bandwidth-delay constrained path selection under inaccurate state information. *IEEE/ACM Transactions on Networking*, Volume: 11, Issue: 3 , Pages:384 – 398, June 2003.

[KORK03b] T. Korkmaz and M. M. Krunz. Routing multimedia traffic with QoS guarantees. *IEEE Transactions on Multimedia*, Volume: 5, Issue: 3, Pages:429 – 443, Sept. 2003.

[KUEH63] A. A. Kuehn and M. J. Hamburger. A Heuristic Program for Locating Warehouses. *Management Science* 9, 643-666, 1963.

[KUIP02] F. Kuipers, P. V. Mieghem, T. Korkmaz, and M. Krunz. An overview of constraint-based path selection algorithms for QoS routing. *IEEE Communications Magazine*, Volume: 40, Issue: 12, Pages:50 – 55, Dec. 2002.

[KUIP03] F. A. Kuipers and P. V. Mieghem. The impact of correlated link weights on QoS routing. *IEEE INFOCOM,* Volume: 2, Pages:1425 – 1434, 30 March-3 April 2003.

[LEE00] W. Lee and B. G. Lee. Pre-computation based selective probing (PCSP) scheme for distributed QoS routing. *IEEE Global Telecommunications Conference (GLOBECOM)*, Volume: 3, Pages:1778 - 1782, 27 Nov.-1 Dec. 2000.

[LEE01] W. Lee and B. G. Lee. Pre-computation based selective probing (PCSP) scheme with imprecise state information. *IEEE International Conference on Communications* (*ICC)*, Volume: 3, Pages:674 – 678, 11-14 June 2001.

[LEE03] S. S. Lee, S. Das, G. Pau, and M. Gerla. Hierarchical multipath approach to QoS routing: performance and cost evaluation. *IEEE International Conference on Communications (ICC)*, Volume: 1, Pages:625 – 630, 11-15 May 2003.

[LI99] B. Li, M. J. Golin, G. F. Italiano, X. Deng, and K. Sohraby. On the optimal placement of web proxies in the Internet. In *Proceedings of IEEE INFOCOM*, Volume: 3, Pages:1282 – 1290, 21-25 March 1999.

[LUI00] K. Lui, K. Nahrstedt, and S. Chen. Hierarchical QoS routing in delay-bandwidth sensitive networks. In *Proceedings of 25th Annual IEEE Conference on Local Computer Networks (LCN)*, Pages:579 – 588, 8-10 Nov. 2000.

[LUND94] C. Lund and M. Yannakakis. On the hardness of approximating minimization problems. *Journal of ACM*. 41, 5 (September), pp. 960-981, 1994.

[MA96] Q. Ma, P. Steenkiste and H. Zhang. Routing High-bandwidth Traffic in Max-min Fair Share Networks. In *Conference proceedings of ACM SIGCOMM'96*, pp 206-217, Stanford, California, August, 1996.

[MA97] Q. Ma and P. Steenkiste. On path selection for traffic with bandwidth guarantees. In *Proceedings of International Conference on Network Protocols*, Pages:191 – 202, 28-31 Oct. 1997.

[MAGO02] D. Magoni and J.-J. Pansiot. Evaluation of Internet topology generators by power law and distance indicators. *10th IEEE International Conference on Networks (ICON)*, Pages:401 – 406, 27-30 Aug. 2002.

[MAHD02] M. Mahdian, Y. Ye and J. Zhang. Improved Approximation Algorithms for Metric Facility Location Problems. In *Proceedings of the 5th International Workshop on Approximation Algorithms for Combinatorial Optimization*. September 2002.

[MEDI00] A. Medina, I. Matta, and J. Byers. On the origin of power laws in Internet topologies. *ACM SIGCOMM Computer Communication Review*, Volume 30, Issue 2, April 2000.

[MEDI01] A. Medina, A. Lakhina, I. Matta, and J. Byers. BRITE: an approach to universal topology generation. In *Proceedings of the Ninth International Symposium on Modeling, Analysis and Simulation of Computer and Telecommunication Systems*, Pages:346 – 353, 15-18 Aug. 2001.

[MIRC90] P. B. Mirchandani and R. L. Francis. *Discrete location theory*. John Wiley & Sons, Inc, 1990.

[MOY94] J. Moy. OSPF Version 2. *RFC 1583*, Internet Engineering Task Force, March 1994.

[NICH98] K. Nichols, S. Blake, F. Baker and D. Black. Definition of the Differentiated Services Field (DS Field) in the IPv4 and IPv6 Headers.*RFC 2474*, Internet Engineering Task Force, December 1998.

[QIU01] L. Qiu, V. N. Padmanabhan, and G. M. Voelker. On the placement of Web server replicas. In *Proceedings of IEEE INFOCOM*, Volume: 3, Pages:1587 – 1596, 22-26 April 2001.

[REKH95] Y. Rekhter and T. Li. A Border Gateway Protocol 4 (BGP-4). *RFC 1771*, Internet Engineering Task Force, March 1995.

[SHMO97] D. B. Shmoys, E. Tardos and K. I. Aardal. Approximation algorithms for facility location problems. In *proceedings of the 29th Annual ACM Symposium on Theory of Computing*. ACM, New York, 265-274, 1997.

[SIAC03] S. Siachalou and L. Georgiadis. Efficient QoS routing. *IEEE INFOCOM*, Volume: 2 , Pages:938 – 947, 30 March-3 April 2003.

[SIGA03] G. Siganos, M. Faloutsos, P. Faloutsos, and C. Faloutsos. Power laws and the AS-level Internet topology. *IEEE/ACM Transactions on Networking*, Volume: 11, Issue: 4, Pages:514 – 524, Aug. 2003.

[SOBR02] J. L. Sobrinho. Algebra and algorithms for QoS path computation and hop-by-hop routing in the Internet. *IEEE/ACM Transactions on Networking*, Volume: 10, Issue: 4, Pages:541 – 550, Aug. 2002.

[STAL02] W. Stallings. High-Speed Network and Internets Performance and Quality of Service, Second Edition. *Prentice-Hall*, Upper Saddle River, New Jersey, 2002.

[TANG02] H. Tangmunarunkit, R. Govindan, S. Jamin, S. Shenker, and W. Willinger. Network topology generators: degree-based vs. structural. In *Proceedings of the 2002 conference on Applications, technologies, architectures, and protocols for computer communications, ACM SIGCOMM Computer Communication Review*, Volume 31, Issue 4, August 2002.

[TAUR01] S. L. Tauro, C. Palmer, G. Siganos and M. Faloutsos. A simple conceptual model for the Internet topology. *IEEE Global Telecommunications Conference (GLOBECOM)*, Volume: 3, Pages:1667 – 1671, 25-29 Nov. 2001.

[THOR03] M. Thorup. Quick and good facility location. In *Proceedings of the fourteenth annual ACM-SIAM symposium on Discrete algorithms*, January 2003.

[WANG02] J. Wang and K. Nahrstedt. Hop-by-hop routing algorithms for premium-class traffic in DiffServ networks. In *Proceedings IEEE INFOCOM*, Volume: 2, Pages:705 – 714, 23-27 June 2002.

[WANG95] Z. Wang and J. Crowcroft. Bandwidth-delay based routing algorithms. *Global Telecomunications Conference, 1995*. GLOBECOM '95., IEEE, Volume: 3, 13-17 Nov. 1995.

[WANG96] Z. Wang and J. Crowcroft. Quality-of-service routing for supporting multimedia applications. *IEEE Journal on Selected Areas in Communications*, Volume: 14 , Issue: 7, Pages:1228 – 1234, Sept. 1996.

[WATT98] D.J. Watts and S.H. Strogatz. Collective dynamics of "small-world" networks. *Nature*, 393, 440-442, 1998.

[WINI03] J. Winick and S. Jamin. *University of Michigan Technical Report CSE-TR-456-02*, http://topology.eecs.umich.edu/inet/

[XIAO02] L. Xiao, K. Lui, J. Wang, and K. Nahrstedt. QoS extension to BGP. In *Proceedings of 10th IEEE International Conference on Network Protocols*, Pages:100 – 109, 12-15 Nov. 2002.

[ZIPF32] G. K. Zipf, Selective Studies and the Principle of Relative Frequency in Language, 1932

[ZEGU96] E. W. Zegura, K. Calvert and S. Bhattacharjee. How to Model an Internetwork. In *Proceedings of IEEE INFOCOM*, 1996.