Collaborative Communications Among Multiple Points

ZHANG Xinyan

A Thesis Submitted in Partial Fulfilment
of the Requirements for the Degree of
Master of Philosophy
in
Information Engineering

June 2004

Acknowledgement

Many people contribute their time and energy in my two-year life as a master of philosophy student in CUHK. I would like give my thanks to them:

The supervisor, surely should be the first one to thank sincerely: Professor Peter Tak-Shing Yum. He has given me the opportunity to perform research under his supervision. His advices are very genuine and valuable for me. I can feel that he has tried his best and done everything possible to help me. His attitude to research and life influences me greatly. The most valuable point I should learn from him is that he usually knows which things should be tackled seriously and which not, where I mostly cannot distinguish.

Grateful thanks to the friends in the office who constitute the environment of my life largely. Though everybody should do the job independently, there are still many useful discussions. Your existences have given me many good feelings (at least I can find someone to eat together :). The time with all of you is undoubtedly an important part of my life.

I would like to thank staffs in the departments of information engineering for their assistances and advices.

Finally, I would like to thank my parents, you are always my strongest supporters.

摘要

近來，多用戶合作通訊的問題引起了廣泛的研究興趣。這些問題包括：點對點通訊架構，應用層組播以及應用層路由。在這篇論文中，我們主要研究一下兩個相關問題。

- 爲了讓網絡有更好的數據傳輸，我們提出一種應用層多路徑傳輸架構 —— MultiServ。据我們所知，MultiServ 是第一個定量對應用層多路徑傳輸進行優化的技術。我們的貢獻有：完整的應用層覆蓋網絡構建和分佈式多路徑路由協議 。經過構造應用層覆蓋網絡和應用分佈式多路徑路由協議 ，在 MultiServ 平臺上可以完成很多原先在 IP 網絡中很困難的任務。例如大範圍的媒體流和應用層組播。同時，網絡運營商(ISP)也可以使用 MultiServ 來提供更好的控制數據傳輸。和以前的相似方法相比，MultiServ 適用範圍更大而且易於實現。仿真試驗告訴我們 (1) 和最短路徑和負載平衡多路徑方法 比較，MultiServ 能夠使傳輸速率加倍，從而接近可能的最優方法；(2) MultiServ 可以在大範圍網絡使用，其性能高效而穩定；(3) MultiServ 能夠對快速自適應各種網絡異常情況 。我們也在環球網絡平臺 (Planet Lab)上簡要測試了 MultiServ。
- 我們基於覆蓋網絡 提出一種新的應用層組播架構 ——動態分佈式媒體流 (Dynamic Distributed Streaming, DDS)。我們比較了 DDS 和基於樹結構的應用層組播的數據丢失。結果顯示 DDS 在用戶動態（加入退出頻繁）的情況下性能好很多。在用戶較多的情況下，DDS 的數據丢失只有樹形結構的 10%。

Abstract

Recently, there are research interests in the problem of collaborative communications among multiple users, such as the peer-to-peer overlay architecture, application layer multicast and application layer routing. This thesis addressed the following two problems.

- An application-layer multiple path routing architecture called MultiServ is proposed to better distribute traffic. To our knowledge, MultiServ is the first quantitative approach toward optimal distributed routing. We provide the complete set of application-layer overlay construction and distributed multiple path routing schemes. By building efficient overlay network and using distributed routing strategies and its implementations, MultiServ enables services such as large scale streaming and application layer multicasting to be more smoothly executed on the current IP network. MultiServ can also be used in ISP level to provide better traffic management by aggregation and rerouting. Comparing with conventional traffic engineering methods, MultiServ is more scalable and easier to deploy. Simulation results show that (i) the MultiServ routing strategies can double the throughput when compared to shortest path routing and equal loading multipath routing and near optimal performance compared with

centralized traffic engineering in reasonable traffic load; (ii) MultiServ can be deployed in large scaled overlays with efficient and stable performance; (iii) MultiServ has a quick response to traffic change and can adapt the capacity variations in real environment. These performance gains are also briefly demonstrated in planet-lab experiments.

- A new framework called dynamic distributed streaming (DDS) is presented for both on-demand streaming and live-streaming using overlay network. A user model is built and the streaming data outage is derived and compared with application-layer multicast. Results show that in dynamic user environment DDS can perform much better than existing approaches. In large overlays, the data outage in DDS can be as low as 10% compared to that in tree based application layer multicast.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

1.1 Multiple Point Communication

Multiple point communication means direct communication among many users. Numerous applications are using the multiple-point communication model, including video conferencing, network gaming and content distribution. In the client and server model, a client mainly communicates with the server, even client to client communication is sometimes performed indirectly through the server. Obviously, the server may become a bottleneck in this model. Therefore, multiple point communication usually faces the problem of performance, scalability and stability in current communication model.

An example illustrates multiple point communication. Suppose a user want to share a video clip to other users. One common way is to upload it to a server so that everybody can download or stream it from the server. However, if the video clip is very interesting and draws many peoples' attention so that many users would download or stream it in a short period of time (this phenomenon

is called *flash crowd*), the bandwidth and processing power of the server may be exhausted and users may have bad experience retrieving the video clips, or even the server may be crashed and thus the clip will be unavailable. To avoid this situation, an alternative for distributing the video clips is to utilize the end-to-end collaborative communications. For example, the source can upload the clips to two users and each user having the clips (or partial clips) can upload to another two users. This application layer multicast provides much better performance and scalability. However since the machines of users are not as stable as server and user can leave the system at any time, without a robust protocol, the service is unreliable.

From the above example, we see that a collaboration model is essential for multiple point communication. The key issue is how to allocate the resources, such as bandwidth, processing power and storage, so that the system can perform better. This is the focus of this thesis.

1.2 Major Contributions

The contribution of this thesis can be described in the following two areas.

1. MultiServ

   An application-layer multiple path routing architecture called MultiServ is proposed to better distributing traffic. To our knowledge, MultiServ is the first quantitative approach toward optimal distributed routing scheme.

   We propose a complete set of application-layer overlay construction and distributed multiple path routing schemes. By building efficient overlay

network and using distributed routing strategies and its implementations, MultiServ enables services such as large scale streaming and application layer multicasting to be more smoothly executed on the current IP network. MultiServ can also be used in ISP level to provide better traffic management by aggregation and rerouting.

Comparing with conventional traffic engineering methods, MultiServ is more scalable and easier to deploy. Simulation results show that (i) the MultiServ routing strategies can double the throughput when compared to shortest path routing and equal loading multipath routing and near optimal performance compared with centralized traffic engineering in reasonable traffic load; (ii) MultiServ can be deployed in large scale networks with efficient and stable performance; (iii) MultiServ has a quick response to traffic change and can adapt the capacity variations in real environment. These performance gains will be briefly demonstrated in planet-lab experiments.

2. Distributed Dynamic Streaming

   A new framework called dynamic distributed streaming (DDS) is presented for both on-demand streaming and live-streaming using overlay network. A user model is built and the streaming data outage is derived and compared with application-layer multicast. Results show that DDS can perform much better in dynamic user environment. In large overlay networks, the data outage in DDS can be as low as 10% compared to that in tree based application layer multicast.

1.3 Thesis Organization

This thesis is organized as follows. In chapter 2, we discuss the related work on multiple point communications, including peer-to-peer networks, application layer multicast and Internet traffic engineering. In chapter 3, an application-layer multiple path routing architecture called MultiServ is proposed to provide better end-to-end communication performance and enable more services for end hosts. Chapter 4 introduces a new framework called dynamic distributed streaming for both on-demand streaming and live-streaming using overlay network. In chapter 5, we summarize those approaches.

Chapter 2

Related Work

2.1 Peer-to-Peer Networks

Peer-to-peer network (P2P) [69] is characterized by direct access between peer computers, rather than indirectly access through a centralized server. File sharing is a dominant P2P application on the Internet, allowing users to easily contribute, search and obtain content.

P2P file sharing architectures can be classified by their "degree of centralization", i.e. to what extent they rely to one or more servers to facilitate the interaction between peers. P2P filesharing architecture can be classified into three categories:

- **Hybrid decentralized architectures**

  Napster [26] is the first generation of P2P file sharing program. Napster uses a hybrid decentralized architecture. It is doubtful that Napster is not real P2P systems. In Napster, there is a central server facilitating the interaction between peers by maintaining directories of the shared files

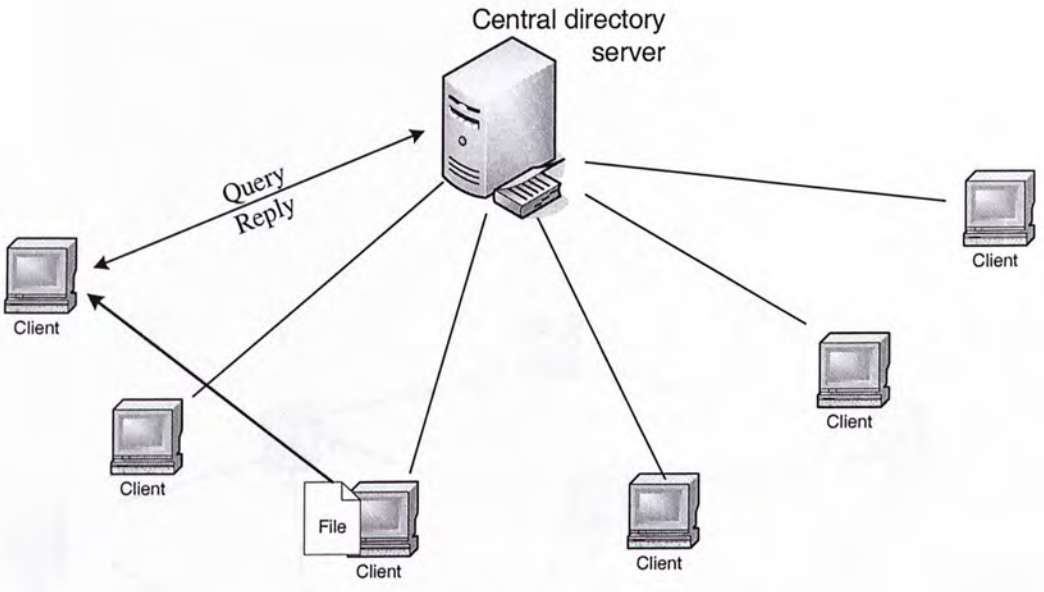


Figure 2.1: Architecture of Napster.

stored on the respective hosts of registered users to the network. These central servers will perform the lookups and identifying the nodes of the network (i.e. the computers) where the files are located. Two peer clients make end-to-end connections to transfer the files. Figure 2.1 illustrates the architecture of Napster.

- **Purely decentralized P2P architectures**

  Gnutella [25] and Freenet [70] [71] utilize purely decentralized P2P architectures. All nodes in the network perform exactly the same tasks, acting as both servers and clients, and there is no central coordination of their activities. The nodes of such networks are termed "servents" (SERVers+clieENTS). Figure 2.2 illustrates the architecture of Gnutella.

- **Partially centralized systems**

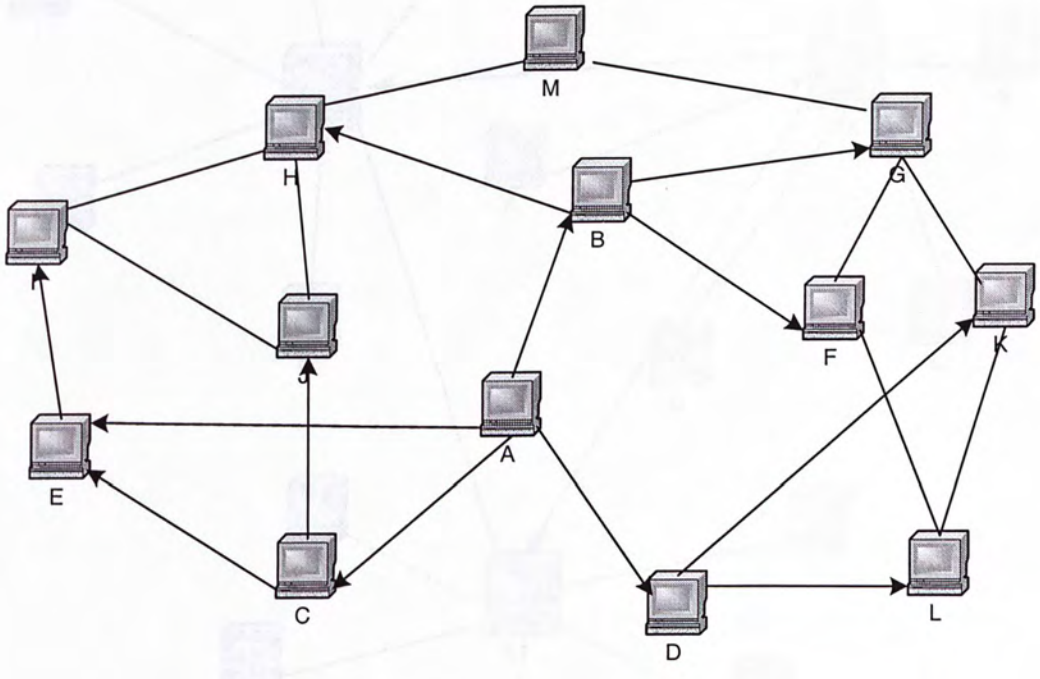


Figure 2.2: Architecture of Gnutella. No central directory server maintains an index of the metadata. So random search is essential. A search request from node A with TTL = 1 reaches node B, C, D and E, with TTL = 2 reaches node B to L, however, cannot reach node M until TTL = 3.

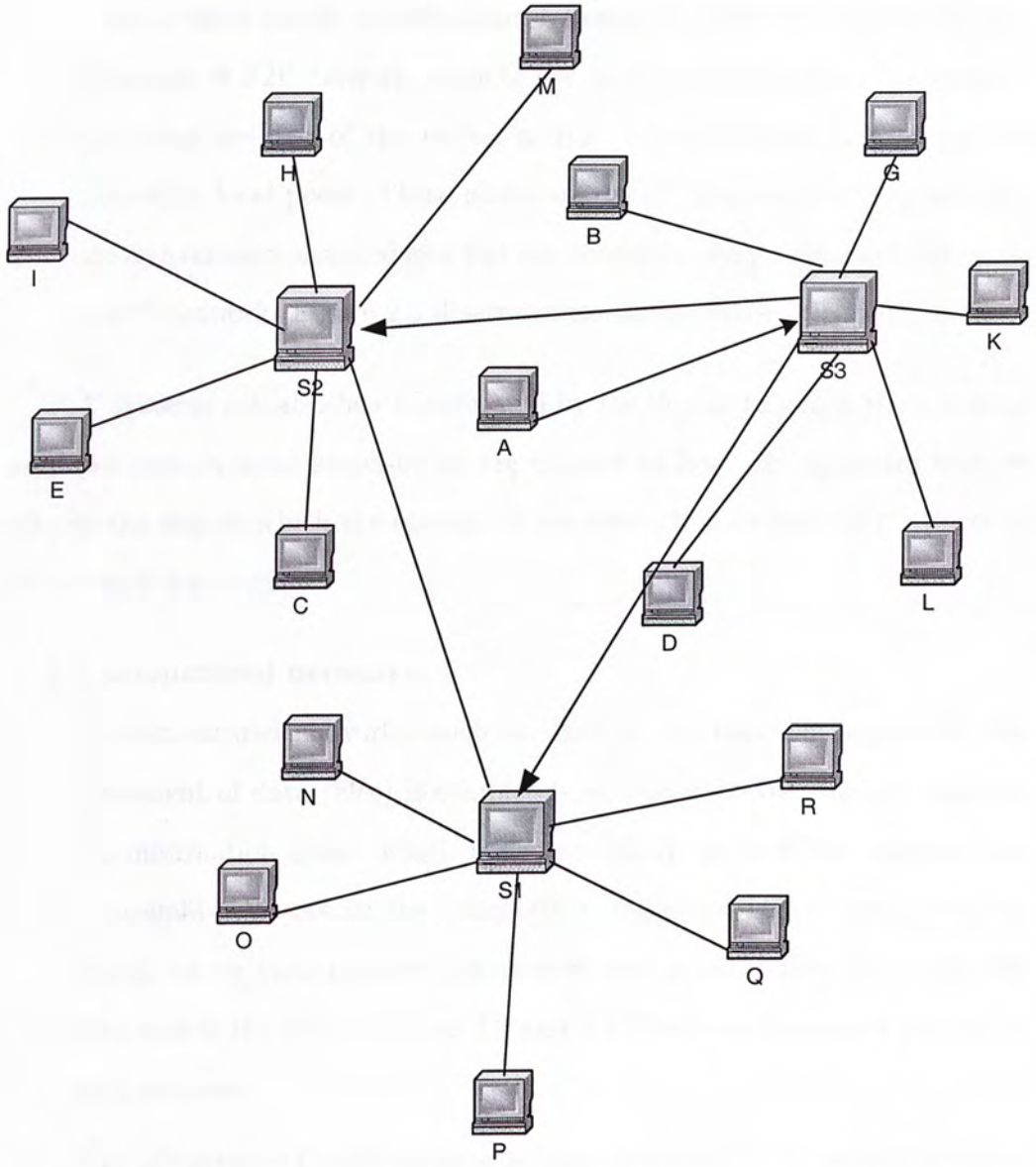


Figure 2.3: Architecture of Kazaa. No central directory server maintains an index of the metadata. A search request from node A is first sent to supernode $S3$, supernode $S3$ will forward the search request to other supernodes $S1$ and $S2$.

Kazaa [27] is an example of partially centralized systems. The basis is the same as with purely decentralized systems. In order to enhance the performance of P2P network, some of the nodes assume a more "important" role than the rest of the nodes, acting as local central indexes for files shared by local peers. These nodes are called "Supernodes". Supernodes are dynamically assigned and will not constitute single points of failure for a P2P network. Figure 2.3 illustrates the architecture of Kazaa.

P2P systems can also be differentiated by the degree to which these overlay networks contain some structure or are created ad-hoc. By structure here we refer to the way in which the content of the network is located with respect to the network topology.

- **Unstructured networks**

  In unstructured networks (such as those in the Gnutella approach), the placement of data (files) is completely unrelated to the overlay topology. No information about which nodes are likely to have the relevant files is available. To obtain the information, the client will launch a random search where various nodes are probed and asked if they have any files that match the query. Figure 2.2 and 2.3 illustrate the search process in such systems.

  The advantage of such systems is that they can easily accommodate a highly transient node population. The disadvantage is that the queries cannot be distributed widely, and therefore it is hard to find the desired files, in particular, sparse files. For this reason unstructured P2P systems are considered to be unscalable [72].

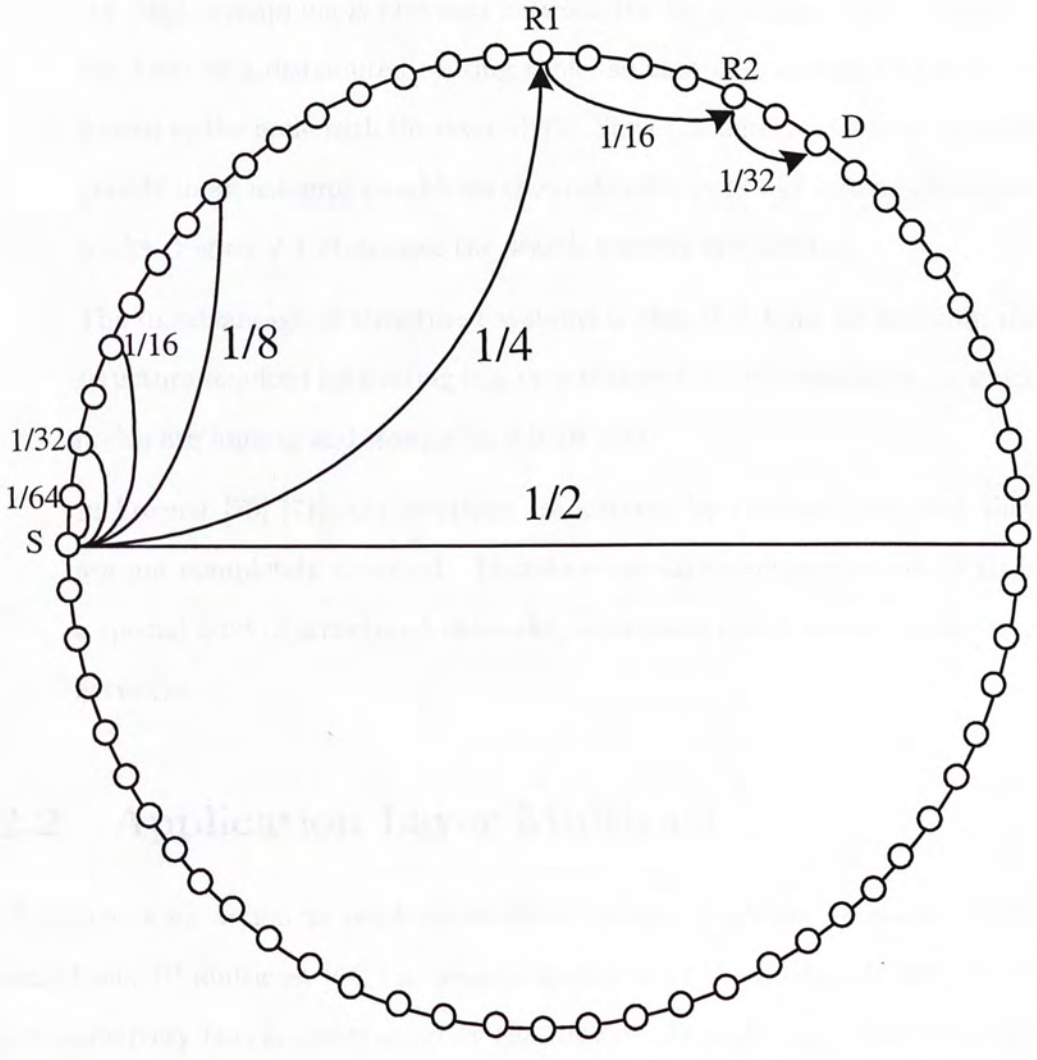


Figure 2.4: Architecture of Chord. No central directory server maintains an index of the metadata. Every node maintains a routing table. A search request from node S is sent through node $R1$ and $R2$, and finally get the search response from nodes D. The length of search path is $O(\log N)$ (N is the number of nodes).

- **Structured networks**

  In structured networks (such as Chord [49], CAN [50], Pastry [54], Tapastry [56]), a mapping is provided between the file identifier and location, in the form of a distributed routing table, so that queries can be efficiently routed to the node with the desired file. Structured networks have emerged mainly in an attempt to address the scalability issues in unstructured networks. Figure 2.4 illustrates the search process in Chord.

  The disadvantage of structured systems is that it is hard to maintain the structure required for routing in a very transient node population, in which nodes are joining and leaving at a high rate.

  In Freenet [70] [71], file locations are affected by routing hints, but they are not completely specified. Therefore not all searches succeed. This is a special kind of structured networks, sometimes called loosely structured networks.

2.2 Application Layer Multicast

Multicast is an action to send information to more than one receivers at the same time. IP multicast [44] has been proposed more than a decade ago, where a data delivery tree is constructed by the routers. As multicast packets flow on this tree, they are appropriately replicated by the routers at the different branch points of the tree. IP multicast is the most efficient way to perform group data distribution, as it is able to reduce packet replication on the wide-area network to the minimum necessary.

However, many Internet service providers (ISPs) are still reluctant to provide wide-area multicast routing service [58], and deployment of IP multicast has been limited and sparse due to a variety of technical and non-technical reasons. Specifically, multicast is still a hot and complex research subject, many protocols are not yet finalized, and monitoring IP multicast is not easy. For example, IP Multicast requires routers to maintain per group state(and in some proposals per source state in for each multicast group). The routing and forwarding table at the routers now need to maintain an entry corresponding to each unique multicast group address. However, unlike unicast addresses, these multicast group addresses can not be easily aggregated. Also, multicast breaks the traditional pricing model where only the incoming flow is charged, and the pricing model for multicast traffic is not yet well-defined. Further more, the use of multicast is still driven more by the academic community than by customer demand. The ISPs seek for alternative solutions rather than deploy expensive IP multicast .

Therefore Application layer multicast is proposed as an alternative technique for multicasting. As the name suggests, in application layer multicast, the multicasting functionality is implemented at the application layer, i.e. at the end-hosts instead of the network routers.

Figure 2.5 illustrates the differences of several multicast scenarios. In the figure, A is the source and B, C and D are the receivers. In IP multicast, data packets are replicated at routers inside the network illustrated (see 2.5 (a)). Clearly IP multicast is the most efficient way to perform group data distribution where duplicated transmission is eliminated. In places where multicast service is not available, multiple unicasts from the source can emulate the multicast

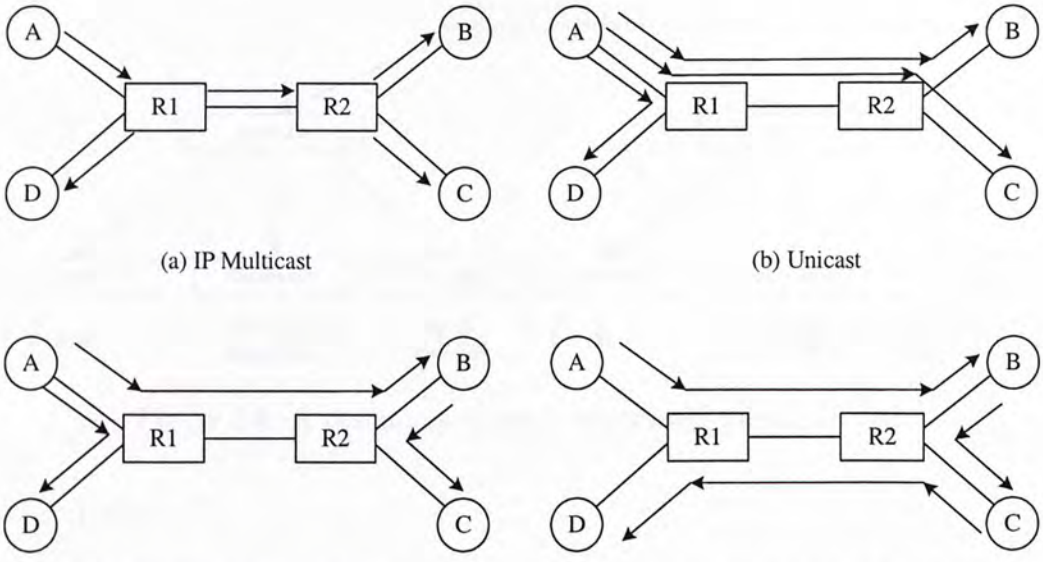


Figure 2.5: Multicast scenarios. Square nodes are routers, and circular nodes are end-hosts.

function (see 2.5 (b)). However we can see three duplicated flows in link A-R1 and two in link R1-R2. In application multicast scenario illustrated in Figure 2.5 (c) and (d), We can see that not only the source but also receivers can deliver the data to other members in multicast.

Several performance metrics have been defined to characterize application layer multicast performance and impacts on the network, as described below:

Stress [18] is defined as the number of identical packets carried on a link. The optimal value, achieved with native multicast routing, is of course 1.

Stretch is also called *relative delay penalty* in [18], the stretch metric between a source and a destination member is the ratio of the delay between them along the overlay distribution topology, to the delay of the direct unicast

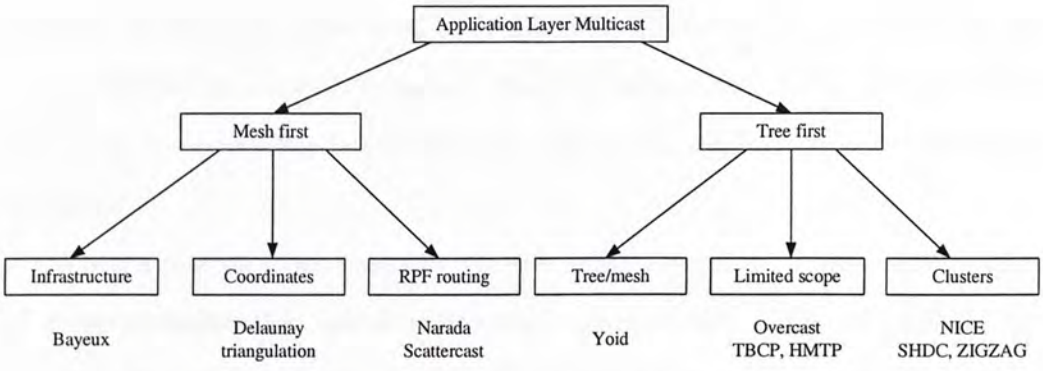


Figure 2.6: A taxonomy of application layer multicast [64].

path.

Control overhead is the cost of Maintaining the application layer multicast topology. The cost includes control information exchanged (number of messages processed and bandwidth).

Figure 2.6 classifies the application layer multicast proposals according to different building algorithm. Application layer multicast methods differ in the way they create the overlay topology: some of them create the tree topology first, while others create a mesh topology first.

The typical "mesh first" approach is Narada [18], the proposals that assign an arbitrary coordinate to each member and then perform Delaunay triangulation [65], and Bayeux [66].

The Narada protocol was one of the first application layer multicast protocols that demonstrated the feasibility of implementing multicast functionality at the application-layer. Narada defines a special designated host, called the Rendezvous Point (RP), that is used to boot-strap the join procedure of a new

member. In fact, all application layer multicast protocols use an entity equivalent to the RP in Narada to initiate the join mechanism. In the thesis, we use this term to denote the boot-strapping host for all application layer multicast protocols.

When a new member wants to join the multicast group, it first obtains a list of group members that are already joined into the mesh. This information can typically be obtained from the RP, which maintains state about all members joined to the multicast group. The new member then randomly selects a subset of these members and attempts to join the mesh as neighbors of these members. The join procedure succeeds when at least one of these members accept the new member as its mesh neighbor.

After joining the mesh, the new member starts exchanging periodic refresh messages with its mesh-neighbors. Whenever a new member joins or an existing member leaves the group (and the mesh), this group change information is propagated through the mesh to all the other members.

The members of the group run a routing protocol to compute unicast paths between all pair of members on the mesh. The multicast data delivery path with any specific member as the source can then be computed using the well-known Reverse Path Forwarding check employed by IP multicast protocols (e.g. DVMRP [20]).

The data delivery paths in Narada are spanning trees of the mesh. Therefore, the quality of the data delivery path (i.e., the stress and stretch properties) depends on the quality of links that are part of the mesh. When new members join, or when the mesh recovers from partitions, a random set of edges are added to the mesh. Thus, periodic refinements are made to mesh edges to improve the

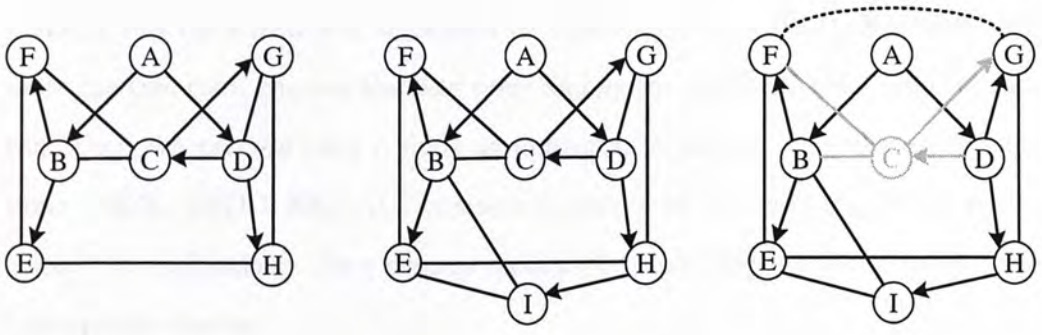


Figure 2.7: Control and data paths in Narada. Neighbors on the control path are connected by edges. Data are transferred through edges with arrow.

quality of data delivery paths. Narada allows for incremental improvement of mesh quality. Members probe each other at random and new links may be added depending on the perceived gain in utility in doing so. Further, members continuously monitor the utility of existing links, and drop links perceived as not useful.

We now show an example of Navada network. In figure 2.7(a), a Navada overlay network is running application layer multicast tasks where node A multicasts data to other node. In figure 2.7(b), A new node I joins the overlay network and selects node B, E and H as its neighbors. Data will be delivered from node H to node I. In 2.7(c), node C leaves and data cannot be delivered from node C to node G. Therefore, data will be delivered from node D to node G instead. New connection from node F to node G will be established to maintain the connectivity of overlay network.

The "tree first" approaches include YOID [45], TBCP [62], HMTP [55], SHDC [63], NICE [10], Overcast [46], and ZIGZAG [61]. Some of them (TBCP,

HMTP) rely on a recursive algorithm to build the tree: a new comer first contacts the tree root, chooses the best node among the root's children, and repeats this top-down process until it finds an appropriate parent. The clustering solutions (NICE, SHDC, ZIGZAG) create a hierarchy of clusters (i.e., sets of nodes "close" to each other). New comers recursively cross this hierarchy to find the appropriate cluster.

Here we briefly introduce the NICE protocol. The NICE protocol arranges the set of members into a hierarchical control topology. As new members join and existing members leave the group, the basic operation of the protocol is to create and maintain the hierarchy. The hierarchy implicitly defines the multicast overlay data paths and is crucial for scalability of this protocol to large groups. The members at the bottom of the hierarchy maintain (soft) state about a constant number of other members, while the members at the top maintain such state for about $O(\log n)$ other members.

The NICE hierarchy is created by assigning members to different levels (or layers) as illustrated in figure 2.8 (a). Layers are numbered sequentially with the lowest layer of the hierarchy being layer zero (denoted by L_0). Members in each layer are partitioned into a set of clusters. Each cluster is of size between k and $3k - 1$, where k is a constant, and consists of a set of members that are close to each other. Further, each cluster has a cluster leader. The protocol distributedly chooses the (graph-theoretic) center of the cluster to be its leader, i.e. the cluster leader has the minimum maximum distance to all other members in the cluster. This choice of the cluster leader is important in guaranteeing that a new joining member is quickly able to find its appropriate position in the hierarchy using a very small number of queries to other members.

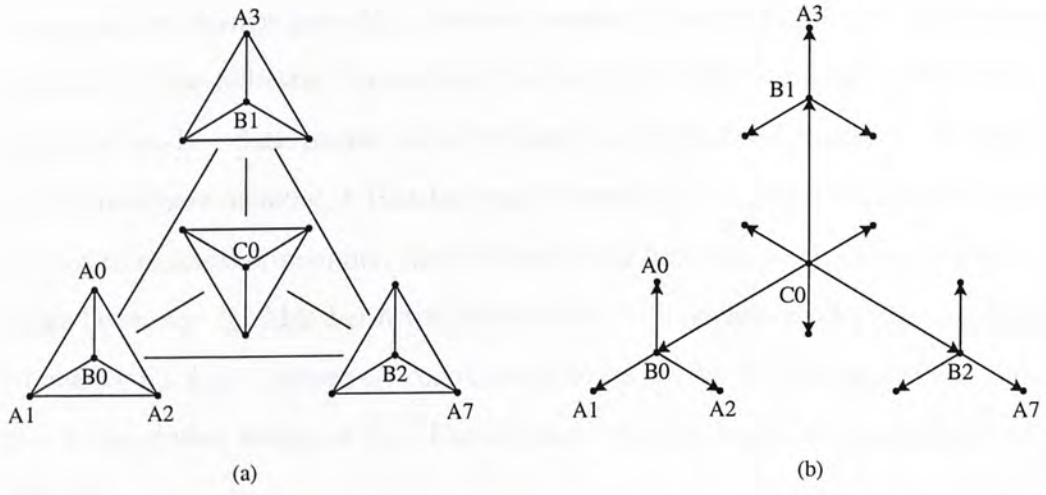


Figure 2.8: NICE hierarchy and control and data topologies for a two-layer hierarchy. All A_i hosts are members of only L_0 clusters. All B_i hosts are members of both layers L_0 and L_1. The only C host is the leader of the L_1 cluster comprising of itself and all the B hosts.

The members are assigned to the different layers as follows: All members are part of the lowest layer, L_0. A distributed clustering protocol at each layer partitions these members into a set of clusters with the specified size bounds. The protocol also chooses the member which is the graph theoretic center of the cluster, to be the leader of the cluster. The cluster leaders of all the clusters in layer L_i join layer L_{i+1}.

A new member joins a L_0 cluster that is closest to itself with respect to the distance metric. Locating this L_0 cluster is approximated by a sequence of refinement steps, where the joining member starts with the topmost layer and sequentially probes one cluster in each layer to find the "closest" member in that layer.

The member hierarchy is used to define both the control and data overlay topologies. In the control topology, all members of each cluster peer with each

other and exchange periodic refreshes between them. The data topology is defined by the following forwarding rule on the control topology: The source member sends a data packet to all its peers on the control topology. Consider an intermediate member, h that belongs to layers $L_0 \cdots L_j$ that receives the data packet from another member, say p. Then p and h belong to the same cluster in some layer, say L_i. Member h will forward the data packet to all other members of cluster $C_k, k \neq i$ (where C_k corresponds to its cluster in layer L_k) if and only if h is the cluster leader of C_k . The ensuing data topologies are shown in figure 2.8 (b).

2.3 Internet Traffic Engineering

Major objectives of Internet traffic engineering are to enhance the performance of an operational network at both the traffic and resource levels. To achieve this, the main task of Internet traffic engineering is addressing traffic oriented performance requirements, while utilizing network resources economically and reliably. the Internet has been a best effort service environment until recently. In particular, very limited traffic management capabilities existed in IP networks to provide differentiated queue management and scheduling services to packets belonging to different classes.

One of the most significant functions performed by the Internet is the routing of traffic from ingress nodes to egress nodes. Therefore, one of the most distinctive functions performed by Internet traffic engineering is the control and optimization of the routing function, to steer traffic through the network in

the most effective way. Internet has employed distributed protocols for intra-domain routing. These protocols are highly scalable and resilient. However, they are based on simple algorithms for path selection which have very limited functionality to allow flexible control of the path selection process.

Currently, service providers apply many of the traffic engineering mechanisms to optimize the performance of their IP networks. These techniques include capacity planning for long time scales, routing control for medium time scales, the overlay model also for medium time scales, and traffic management mechanisms for short time scale.

When a service provider plans to build an IP network, or expand the capacity of an existing network, effective capacity planning should be an important component of the process. Such plans may take the following aspects into account: location of new nodes if any, existing and predicted traffic patterns, costs, link capacity, topology, routing design, and survivability.

Interior gateway protocols (IGPs), such as Intermediate System - Intermediate System (IS-IS [41]) and Open Shortest Path First (OSPF [40]), commonly used to route traffic within autonomous systems in the Internet, are topology-driven and employ per-packet progressive connection control. Each router makes independent routing decisions using a local instantiation of a synchronized routing area link state database. Route selection is based on shortest path computation using simple additive link metrics. In that case, traffic engineering with IGP is done by increasing the OSPF or IS-IS metric of a congested link until enough traffic has been diverted from that link. Modifying IGP metrics to control traffic routing tends to have network-wide effect. Consequently, undesirable and unanticipated traffic shifts can be triggered. Recently, some new

intra-domain traffic engineering methods have been proposed [38, 13, 15]. These methods take traffic matrix, network topology, and network performance objectives as inputs, and produce load-sharing ratios to be set at the head-end routers of some Equal Cost Multiple Paths (ECMPs) as outputs. These new progresses open new possibility for intra-domain traffic engineering with IGP to be done in a more systematic way.

Overlay traffic engineering was proposed to circumvent some of the limitations of IP systems. The basic idea is to introduce a secondary technology, such as ATM, with virtual circuit and traffic management capabilities, into the IP infrastructure in an overlay configuration. The virtual circuits of the secondary technology serve as point-to-point links between IP routers. With this approach, service providers can establish logical connections between the edge nodes of a backbone, and overlay them onto the physical topology.

MultiProtocol Label Switching (MPLS) is an advanced forwarding scheme which also includes extensions to conventional IP control plane protocols. Recent developments in MPLS [31, 32, 33, 34, 35, 36, 37] open new possibilities to address some of the these "IP limitation". A framework for MPLS is presented in [34] and an architecture is proposed in [37]. The requirements for traffic engineering over MPLS were articulated in [31]. Although MPLS is a relatively simple technology (based on the classical label swapping paradigm), it enables the introduction of sophisticated control capabilities that advance the traffic engineering function in IP networks [31, 32, 33, 35, 36]. A particularly interesting aspect of MPLS is that it can efficiently support origination connection control through explicit label-switched paths. When MPLS is combined with differentiated services and constraint-based routing, QoS provisioning in IP networks

can be realized.

Chapter 3

MultiServ: Application Layer Multiple Path Routing

In this chapter, an application-layer multiple path routing architecture called MultiServ is proposed to better distribute traffic. To our knowledge, MultiServ is the first quantitative approach toward optimal distributed routing. MultiServ's contributions are the complete set of application layer overlay construction and distributed multiple path routing schemes. We provide the complete set of application-layer overlay construction and distributed multiple path routing schemes. By building efficient overlay network and using distributed routing strategies and its implementations, MultiServ enables services such as large scale streaming and application layer multicasting to be more smoothly executed on the current IP network. MultiServ can also be used in ISP level to provide better traffic management by aggregation and rerouting.

3.1 Motivation

Good end-to-end performance in Internet applications requires small delay and sufficient bandwidth. However, these requirements may not be easily satisfied. For example, large scale streaming service often causes link congestion and cannot be easily deployed in large scale. When a link is congested, all transmissions passing through it are affected. Therefore, congestion in a small set of links may affect a large number of end-to-end applications. On the other hand, it was reported that the utilization of Internet backbones is quite small, no more than 30% on links most of the time [1, 2]. The contradiction in bandwidth consumption and supply lead us to investigate better routing practices.

Some causes of congestion in the current Internet are:

- Traffic hot spot causes unbalanced loading in different paths.
- The dominating transport protocol in Internet, TCP, does not perform stably in large delay bandwidth production environment, limiting transmission throughputs.
- The bursty nature of Internet traffic.
- ISPs often use inefficient routing policies to safeguard their own interests.

Various solutions have been proposed for these problems. Over-provision is the method ISPs use to deal with congestion due to growing demand. However, it is expensive to estimate the traffic and deploy sufficient equipments to accommodate the future demand. From research community, improved protocols [3, 4] are proposed to exploit the available bandwidth, which still need time to

validate. There are routing improvements also on inter-domain site, such as the Border Gateway Protocol (BGP [39, 5, 6]).

To solve the problem, we propose an architecture named MultiServ that uses overlay network to provide better Quality of Service to end-hosts. In current Internet, typically every user uses a single direct connection to exchange data with another user. This performs well when there is no congestion in the path to used. However, when congestion occurs, the user has to suffer from the unstable transmission. But there may be users who do not experience congestion in the path to the same destination. In that case, if the original user transmits through this kind of users, using them as a proxy, the transmission may be accelerated. Meanwhile, the traffic can be delivered through other less congested paths so that the load of congested path is alleviated. This is the basic concept of our MultiServ model.

To use the MultiServ architecture, we need cooperation of multiple users. Therefore a special overlay should be constructed to cooperatively deliver traffic. Each host in the overlay can be a sender as well as a receiver. A sender has several neighbors, which are also the hosts selected from the overlay. During the transmission, the neighbors will be responsible for delivering some or all packets from the sender when encountering congestion. To some extent, transmission through the neighbors can be treated as the complement to the traditional end-to-end delivery.

To achieve this, first we use a heuristic method to construct a special overlay network by selecting the appropriate neighbors. Our algorithm selects the neighbor not only on consideration of the path QoS, but also in such a way that

the paths from the sender to the neighbors are maximally disjoint. We then propose distributed routing strategies to balance the utilization of different logical links and enhance the performance of data forwarding. An implementation of the routing strategies uses rate-based congestion control algorithm to dispatch the packets for further transmission. The implementation also minimizes the traffic burst by sending data using a unified rate in a smoothed manner.

Here we show two possible usages of MultiServ model. Ordinary users can benefit from MultiServ model. By building a large scale overlay network using a system service, the user can experience better QoS using MultiServ-aware software. The sending and receiving packets will go through multiple paths, where alternative paths will be used as a complement to direct connection. So the user will not experience worse than using best-effort Internet. This gives the incentive for the user to use that kind of software.

The model can be extended to ISP level as illustrated in Figure 3.1. Typically congestion occurs at the edge of an ISP due to unbalanced traffic to each gateway and traffic burst generated from the users. Using MultiServ, special servers can be placed in ISP gateways to redistribute the traffic to other gateways. The servers will aggregate the traffic sending to outside and deliver packets to other special servers in different ISPs by agreement with other ISPs. MultiServ can adaptively combine capacity of different paths to deliver the data to the destination efficiently using our proposed distributed routing strategies.

Recently the notion of overlay network and application layer protocols has attracted Internet researchers' attentions. For instance, resilient Overlay Networks (RON) [7] was proposed to allow end-hosts and applications to cooperatively gain improved reliability and performance in the Internet in comparison

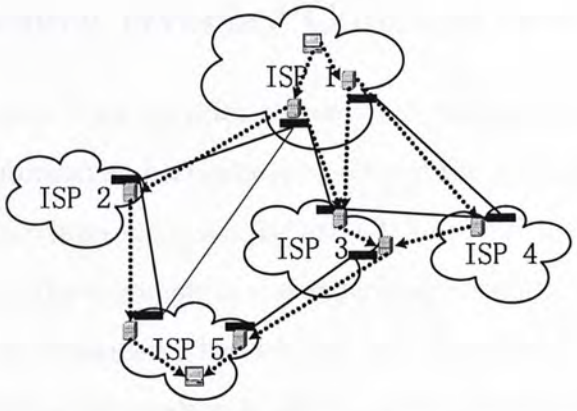


Figure 3.1: MultiServ illustration.

with traditional routing schemes. OverQoS [8] aims to provide a new platform to offer Quality-of-Service (QoS) using overlay network. Service Overlay Networks [9], similar to our approach, purchases bandwidth with certain QoS guarantees from individual network domains via bilateral service level agreement (SLA) to build a logical end-to-end service delivery infrastructure on top of existing data transport networks. Different from the above approach, our approach mainly focuses on the mechanism of overlay construction and routing strategy. We try to generalize the above model to make it efficient, scalable and practical both for ISPs and end-users.

The remainder of the chapter is organized as follows. Section 2 introduces the mechanism to construct the MultiServ overlay. Section 3 presents the routing strategy and implementation in details. Section 4 verifies the model using simulation and real experiments. In section 5, we summarize the approach and propose the future work.

3.2 MultiServ Overlay Construction

Several methods have been provided to construct overlays which try to exploit the topological information, particularly locality in the underlying network. For example, application-layer multicast [10] and CDN network is of this kind, where each node keeps nearby neighbors in the underlying network in order to transfer data efficiently. In MultiServ, besides the path QoS from the sender to its neighbors, topological information is also a very important factor. Different from the above overlays, here the hosts should have minimal common links to neighbors where the congestion of links could less affect the data transfer performance. Therefore, the nearby hosts may not be the proper neighbor sets to deliver the traffic. For example, the hosts which placed in the same ISP may not form a good neighbors set. So, the criteria for good neighbors set should be that:

(a) The paths to the neighbors should have as few common links as possible;

(b) The paths from the sender to neighbors should maintain good QoS.

Under these assumptions, a good set of neighbors of a sender should be in different ISPs, where these ISPs have good connection quality with the ISP the sender belongs to.

To find a good neighbor set which satisfied the criteria, the common-link parameters D is defined as the number of common links between two paths:

$$D(L_1, L_2) = \frac{1}{2} \sum_{l_1 \in L_1} \sum_{l_2 \in L_2} f(l_1, l_2)$$

where

$$f(l_1, l_2) = \begin{cases} 0 & l_1 \neq l_2 \\ 1 & l_1 = l_2 \end{cases}$$

It is necessary to know each link in the path in order to calculate the disjoint parameter D between two paths. Traceroute can be used to obtain the path information. The tool simply sends a packet with desired TTL n which then listens to an ICMP replied from the router n hops away. The IP address of the router can then be obtained.

Using traceroute, we can obtain IP addresses of routers in the entire path. There might be routers which do not return ICMP packets. However we can ignore those routers and consider the path between successive responsive routers. For example, Host A uses traceroute to find that the routers between A and B are $\{A, r_1, r_2, *, *, r_3, B\}$, where * is the router which does not respond to traceroute. We then assume that there are four links between A and B: $A - r_1$, $r_1 - r_2$, $r_2 - r_3$ and $r_3 - B$. If there is another host C with five links between A and C: $A - r_1$, $r_1 - r_2$, $r_2 - r_6$, $r_6 - r_7$ and $r_7 - B$, then $D(L_{AB}, L_{AC}) = 2$.

Using the information, a host can judge whether the paths to the neighbors are disjoint. To find a good neighbor set for a host, the problem can be formulated as following. Given a host x and a potential neighbor set S, where $|S| = n$, L_s is the path from x to the neighbor s in S. The objective is to select n neighbors to form a neighbor set N which minimizes the common links between x to the neighbors. Define the disjoint parameter D_N for neighbor set N

$$D_N = \sum_{n_i \in N} \sum_{\substack{n_j \in N \\ n_j \neq n_i}} D(L_{n_i}, L_{n_j})$$

The neighbor set should be found with minimized disjoint parameter, that

is

$$\min D_N \qquad \forall N \subset S, |N| = n$$

The complexity to compute the problem is $O(|S|^n)$. It is computable with a small n and small potential neighbor set S. However, the neighbor can be any node in the overlay and the potential neighbor set can be very large, and then the computation and probing costs cannot be afforded. Therefore we use a heuristic algorithm A illustrated in Figure 3.2 to find proper neighbors.

From the algorithm A we can see that each time when the host selects a new neighbor, the value of D_N is decreased. In that case, the neighbor set will refine from time to time and hopefully it could form a good neighbor set after several rounds of refinements.

For the hosts which cannot obtain path information such as hosts hiding in a firewall which blocks all ICMP packets, we will use simple algorithm B to find a proper neighbor set. In this case, we do not have enough information to optimize the neighbor set, however we still can use some intuitive method from simple networking information. For example, the hop number between two hosts can help to estimate if the two hosts are within the same ISP. Generally speaking, the more hops between the hosts, the more likely that the two are in different ISPs. Other techniques, such as King [11], can also estimate the distance of gateways and ASes for hosts.

During transmissions, a host can rank neighbors by transmission rate and

[2]In an overlay network, typically there will be a set of booting hosts which are in charge of collecting existing host addresses in current network and tell the newly joined hosts these addresses. Those booting hosts will be known by a new host and this is the first step for a new host to do and the host address it obtained each time can be considered as address of a random host.

Algorithm A: Heuristic

1. Find a random host x^2;
2. If the QoS to host x is satisfied then obtain path information L from the original host to host x;
3. Current neighbor set N, if $|N| < n$, then $N = N + x$, return;
4. For each neighbor $n \in N$ compute

$$M = \min_{n_i \in N}(D_{N-n_i+x}), \quad j = \arg\min_{n_i \in N}(D_{N-n_i+x})$$

5. If $M < D_N$ then $N = N - n_j + x$, return;
6. Goto 1.

Algorithm B: Simple

1. Find a random host h, put it into candidate list;
2. Get a host from candidate list, named x. If candidate list is empty, goto 1;
3. If the QoS to host x is not satisfied goto 6;
4. If x in the same AS with current neighbors goto 6;
5. Current neighbor set N, if $|N| < n$, then $N = N + x$, return;
6. If the QoS to x is better than any current neighbor, named n_j, then $N = N - n_j + x$, return;
7. Find neighbors of x, put them into candidate list;
8. Goto 2.

Figure 3.2: Algorithms for neighbor selection.

delay. Using this information, our algorithm can compare the measured distance among different neighbors for refinement. The addresses of good neighbors can also be stored for future refinement when the host needs a shutdown. Therefore the next selection process may be eliminated by using stored information.

The selection of good neighbors may be different for different types of hosts. For example, some hosts may be connected to a congested gateway so that almost no hosts can be a good neighbor. In this case, increasing the capacity on the gateway is the only way out. However, for hosts that have congestion on some of its physical links to outside, large portion of hosts can be good neighbors.

We also expect to balance the number of neighbors. In our scenario, one host may serve as neighbors for many hosts. In that case, the bandwidth of that specific host would be exhausted. To solve this problem, for ordinary host, a maximum number of neighbors should be defined. Meanwhile each host maintains a host-cache storing local or nearby host addresses. When one host finds one suitable neighbor, the neighbor may be overloaded and not able to process the relaying tasks The overloaded host gives the original host the nearby hosts, which has high probability to be suitable neighbors, i.e., possible candidates. This step can speed up neighbor selection while balancing the load of hosts.

MultiServ is designed as a system service for hosts, so hosts should not frequently join or quit the overlay. Therefore the availability and stability of the overlay can partly be guaranteed. Furthermore, one neighbor is not the only host that is responsible for delivery. Therefore, failure of neighbors or links is not critical. Even if all links or neighbors fail, the delivery can still use direct connection. However, our algorithm selects neighbors which are likely to be in different regions so that situation will occur with low probability.

3.3 MultiServ Routing

In this section we first describe the importance of routing strategy in overlay network, and then briefly review the traditional optimal routing result in IP network. Understanding the difference between IP network and overlay network, finally the MultiServ routing strategies and implementations are proposed.

3.3.1 The importance of routing strategy

Overlay networks such as peer-to-peer file sharing/distribution network have occupied more and more portion of Internet traffic. However, such overlay network brings problems both for the users and network administrators if it is designed without a controlled manner in data transmission.

Most current peer-to-peer file sharing programs use greedy methods to allocate bandwidth to achieve better performance. That is to say, the program may connect as many sources as possible to download the request content, such as in popular file sharing/downloading program edonkey [28] and bittorrent [29]. However, the concurrent and greedy multiple connection behavior of such programs may be considered as deny of service attacks to the gateways and edge routers by the network administrators. Meanwhile, as the program treats the objective to occupy as much bandwidth as possible, it probably violates the bandwidth share of ordinary users. The typical action against the violation is to ban or shape the bandwidth of the programs. To prevent this from happening, the program should present a better control scheme.

In global scale overlay network, another problem is that the user may suffer is the poor performance. Some of the transmissions between hosts may perform

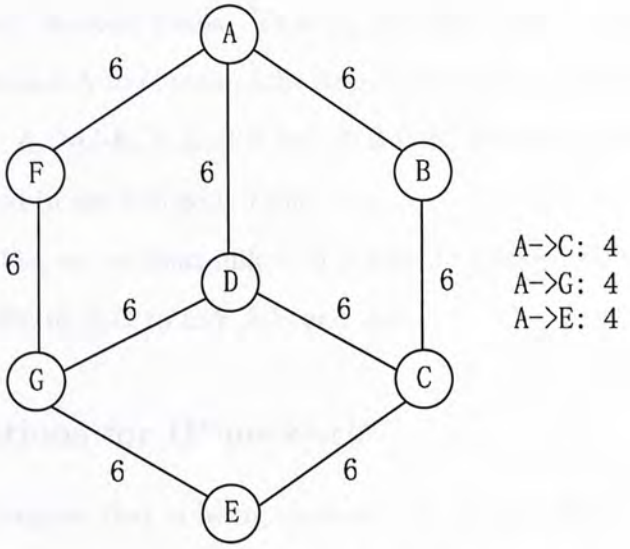


Figure 3.3: Example of routing strategy.

well. However due to link congestion in the underlying network, some of the transmissions may experience low throughput. To achieve better performance in a cooperative environment such as the overlay network, a good routing scheme should be seriously considered.

The above situation suggests that a good routing scheme is expected in current overlay network. Specifically in the MultiServ platform, it is more important, since different routing algorithm will have different effects to underlying network. We illustrate an example in Figure 3.3.

In the figure, each link represents a logical connection between nodes, which has a capacity of 6 units and each demand needs bandwidth of 4 units. We assume the link capacity is bidirectional here for simplicity. To satisfy the traffic demand, there are multiple ways. For example, using fixed single shortest paths to forward data, some of the demands may not be satisfied if two demands share one path. If we use shortest paths data forwarding where the traffic is equally

divided into each shortest paths. That is, demand A to G uses paths A-F-G and A-D-G, demand A to C uses paths A-D-C and A-B-C, demand A to E uses paths A-F-G-E, A-D-G-E, A-D-C-E and A-B-C-E. Thus the utilization of links can be illustrated in the following table 3.1.

From the table, we see that link A-D is heavily loaded and we should move some of the traffic in A-D to link A-B and A-F.

3.3.2 Solutions for IP network

One of the techniques that is being evaluated by many ISPs to achieve better network resource management is traffic engineering. Traffic engineering aims at using information about the traffic entering and leaving the network to optimize network performance. Most often the output of traffic engineering is an "optimal" set of paths and link loads that produce the best possible performance given the available resources. The problems can be formulated as follows.

The IP network can be modeled as a directed graph $G = (V, E)$ where V is the set of nodes and E is the set of links. We assume a traffic matrix T where $T(s_r, t_r) = d_r$ denotes the average intensity of traffic from ingress node s to egress node t for demand $r \in R$. Assume that an optimal allocation based on balanced traffic distribution yields a set of paths P_r for each demand r, so let α be the maximal link utilization for the entire network, c_{ij} be the capacity of link (i, j). The linear program can be formulated as

$$\min \alpha + \epsilon \sum_{(i,j)\in E} \sum_{r\in R} d_r X_{ij}^r$$

s.t.

Table 3.1: Link utilization under bandwidth allocation.

Link	A-B	A-D	A-F	B-C	D-C	D-G	F-G	G-E	C-E
BW	3	6	3	3	3	3	3	2	2
Util	0.5	1.0	0.5	0.5	0.5	0.5	0.5	0.33	0.33

$$\sum_{j:(j,i)\in E} X_{ij}^r - \sum_{j:(i,j)\in E} X_{ij}^r = 0, \quad i \neq s_r, t_r, r \in R$$
$$\sum_{j:(j,i)\in E} X_{ij}^r - \sum_{j:(i,j)\in E} X_{ij}^r = 1, \quad i = t_r, r \in R$$
$$\sum_{r\in R} d_r X_{ij}^r \leq \alpha c_{ij}, \quad (i,j) \in E$$
$$0 \leq X_{ij}^r \leq 1, \quad (i,j) \in E, r \in R$$

where X_{ij}^r is the fraction of traffic for demand r that flows through link (i,j). The objective of the linear programming is to minimize the maximum of link utilization. ϵ is a very small positive number which is introduced so that the optimization not only minimizes α, but also all the X_{ij}^r variables, and ensures that the minimization of α takes higher priority. Solving the linear program gives a traffic allocation $\{X_{ij}^r\}$ that consumes no more than αc_{ij} amount of bandwidth on any link (i,j). It has been shown in [13] that the same performance, in terms of the bandwidth consumed on each link, can be achieved with a set of shortest path with desired weights by solving a dual problem which might simplify the solution to some extent.

Several difficulties exist in deploying the optimal traffic engineering in IP networks. For example, the traffic allocation is in source-destination pair, which may not be quite suitable to deploy in current destination based forwarding routing protocol. Also current IP network cannot support unequal splitting of traffic which is necessary in the optimal traffic engineering. These problems are solved in [14] using destination based aggregation of traffic and approximating

unequal split of traffic using heuristics for traffic splitting, which achieves a near-optimal solutions for current IP networks.

overlay networks do not have the problems of IP networks since overlay networks have the flexibility to use any protocol to communicate with each other. However, the overlay nodes may not perform as stable as routers in IP network. For example, the capacity of logical links and the traffic demand in each node may vary in different time. Also some overlay networks may have much more nodes than routers under an IP network, which increases the complexity of the calculation of optimal solution. Most importantly, the overlay cannot be controlled by a center so that a distributed routing algorithm is expected. Under such a condition, the original optimal solution cannot be used directly and we will present distributed heuristic solutions which have the ability to adapt real network environment for overlay network in the next subsection.

3.3.3 MultiServ routing

The overlay network can be modeled as a directed graph $G = (V, E)$ where V is the set of nodes and E is the set of links.

To perform routing in overlay network, we need a distance matrix $D = [D_{ij}]$, which is the shortest hop distance from node i to node j in the overlay. The distance matrix can be calculated in the following way: a new overlay node broadcasts a distance value message (initialized to zero) to all other nodes through overlay network. Every node the message passes, its distance value is increased by one. Eventually the smallest distance value of received message in each node will be the shortest hop distance to the original node. Nodes can also update distance information with neighbors. In this way the distance matrix D

can be built.

From node i's point of view, the information it knows is the following: c_{ij} is the capacity of logical link $(i,j) \in E$, M_{ik} is the traffic demand to node k, I_{jk} is the traffic input from node j where $(i,j) \in E$ and the destination is node k, D_{ik} is the shortest hop distance to node k and D_j is the distance vector transferred to node i for data forwarding in node j where $(i,j) \in E$.

Given the above information for each node, it is not possible to obtain a global optimization. However, each node may obtain its local optimization in order to satisfy the traffic demand and make the system perform better. The objective of our optimization is to minimize total traffic generated and balance the utilization of logical links while satisfying as much traffic demand as possible. For node i, a linear program can then be formulated:

$$\min \left[\epsilon\alpha + \sum_{k \in V} \sum_{(i,j) \in E} (D_{jk} - D_{ik} + 1) X_{jk} \right] \tag{3.1}$$

s.t.

$$\sum_{(i,j) \in E} X_{jk} = M_{ik}, \quad k \in V \tag{3.2}$$

$$\sum_{k \in V} X_{jk} \leq \alpha c_{ij}, \quad (i,j) \in E \tag{3.3}$$

$$X_{jk} = 0, \quad I_{jk} > 0, (i,j) \in E \tag{3.4}$$

$$X_{jk} \geq 0, \quad I_{jk} = 0, (i,j) \in E \tag{3.5}$$

where X_{jk} is the traffic going through link (i,j) which has the destination k, α is the maximal utilization of the links from i to its neighbors. Constraint (3.2) says that the total flow to node k is M_{ik}. (3.3) indicates that the utilization of each logical link will be less than α. (3.4) prevents sending back the traffic to the node receive from. The objective function (3.1) is to minimize the total

traffic generated. Notice that if all traffic goes through shortest path to the destination, that is through node j where $D_{jk} = D_{ik} - 1$, the generated traffic will be minimized. However not always the traffic demand can be transferred through shortest path in the overlay, therefore traffic demand that may not go through shortest path will introduce extra traffic. Specifically in node i, the extra traffic introduced is

$$T_i = \sum_{k \in V} \sum_{(i,j) \in E} (D_{jk} - D_{ik} + 1) X_{jk}$$

ϵ is a very small positive number which is introduced so that the optimization not only minimizes T_i, but also balances the link utilization, and ensures that the minimization of T_i takes higher priority.

3.3.4 MultiServ routing with bounded complexity

The traffic demand may vary from time to time, so it is necessary to have real time calculation for bandwidth allocation. Although in the routing strategy we proposed, the calculation is affordable. However solving large linear programming is complicated and time consuming, it may be still impossible under real environment. Therefore we propose the following complexity-bounded heuristic method to use in practice.

Notice that to avoid generating extra traffic, the data forwarding should mainly use shortest path. Under this principle, the heuristic method basically uses shortest path first scheduling.

Consider trying to forward as much traffic as possible using shortest path forwarding, we can formulate a maximum flow problem as following.

To maximum the traffic going through shortest path, we use a virtual graph

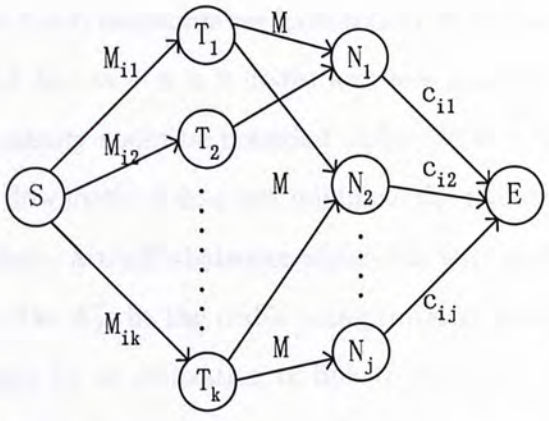


Figure 3.4: Shortest path maximum flow

(G_s, E_s) illustrated in Figure 3.4. In the graph S node is a virtual source which launches all the traffic request in node i. T node denotes traffic demand where the capacity from the source to T_k is M_{ik}, N node denotes the neighbor. The capacity from N_j to the E node is c_{ij}. A link (T_k, N_j) is present if traffic to node k can be forwarded through node j as a shortest path, where $D_{jk} - D_{ik} + 1 = 0$. M is a large value so that the bottleneck of any path from S to E will not be the link which has capacity M.

By solving the maximum-flow problem from S node to E node, we can transfer maximal traffic through shortest path where the flow in (T_k, N_j) represents the traffic going through link (i, j) which has the destination k. However, there might be traffic demand which are still not satisfied. Fortunately, the rest part of the traffic which cannot be forwarded through shortest path can be also scheduled using the similar graph in Figure 3.4. The satisfied part of traffic demand and consumed capacity should be removed from the graph and the structure of the graph also need small modifications, where $(T_k, N_j) \in E_s$ for every $I_{jk} = 0$.

The best known complexity of the maximum-flow algorithm is $O(|V||E|)$.

Suppose each node has n neighbors we have totally m nodes in the overlay, the graph in Figure 3.4 has $m + n + 2$ nodes and less than $(n + 1)m + n$ edges. Therefore the complexity could be bounded under $O((m + n)mn)$.

The maximum flow method does not minimize the maximal value of the link utilization. Therefore, a traffic-balancer algorithm is proposed to balance the traffic in each link. Let X_{jk} be the traffic going through link (i, j) which has the destination k, Define U_j as utilization of link (i, j) where $U_j = \sum_{k \in V} X_{jk}/c_{ij}$. the objective is to minimize the variance of link utilization, defined as follows,

$$V_i = \sum_{(i,j) \in E} \sum_{(i,k) \in E} (U_j - U_k)^2$$

Therefore we propose a heuristic method to decrease the variance iteratively.

The algorithm in Figure 3.5. tries to move traffic from high utilized link to low utilized link while keeping the extra traffic T_i constant. The move of traffic will decrease the objective function V_i each time. This algorithm can be executed until no more traffic could be moved.

3.3.5 Routing implementation

In our model, hosts may use TCP connections to communicate with neighbors. However, the following reasons prevent us from using direct TCP connections between hosts:

- The rate of a TCP connection is not easily controllable;
- Multiple TCP connections may involve the burst of traffic and induce congestion;

Decrease_Pair_Variance(a, b, U)
{
For each Destination x
if $(D_{ax} == D_{bx})$ and $(X_{ax} > 0)$
{
$t = \min(X_{ax}, \sum_{k \in V} X_{ak} - Uc_{ia}, Uc_{ib} - \sum_{k \in V} X_{bk})$;
$X_{ax} = X_{ax} - t$;
$X_{bx} = X_{bx} + t$;
if $(\sum_{k \in V} X_{ak} == Uc_{ia}$ or $\sum_{k \in V} X_{bk} == Uc_{ib})$
return;
}
}

Decrease_Variance()
{
$U = \sum_{(i,j) \in E} \sum_{k \in V} X_{jk} / \sum_{(i,j) \in E} c_{ij}$;
for all neighbor a
for all neighbor b
if $(\sum_{k \in V} X_{ak} > Uc_{ia}$ and $\sum_{k \in V} X_{bk} < Uc_{ib})$
Decrease_Pair_Variance(a, b, U);
}

Figure 3.5: Heuristic algorithm to decrease variance of link utilization

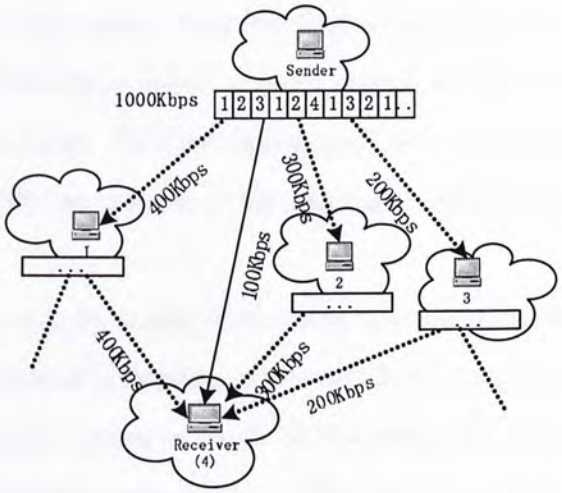


Figure 3.6: Example of joint congestion control.

- TCP is not suitable for applications such as streaming.

For these reasons, it is necessary to propose a TCP friendly congestion control protocol for smooth data transmission with better control.

We propose a rate-based congestion control algorithm, similar with CM [12]. The idea is to aggregate the flows sending from the host, the aggregated flow to one neighbor uses an Additive-increase multiplicative-decrease (AIMD) congestion control in order to be friendly to background TCP flows. The sending rate will increase when no packet loss. Upon a packet loss, the rate will be halved. When persistent congestion occurs, the rate drops to a small value forcing slow start to occur. An ARQ-based mechanism is used. The sender will retransmit the packet until receiving the acknowledgement.

An illustration is presented in Figure 3.6. A large data transmission from the sender to the receiver encountered congestion. Therefore, the sender makes connections to its neighbors to achieve better experience of transmission.

Flows will be aggregated while sending to one neighbor. Each flow may have different transmission speed because neighbors will send out the packets to different destinations. Through aggregation, we can easily adjust the rate of flows by setting different weights in the aggregate flow, as the scheme described in [8].

To further smooth the traffic from hosts, also set up a framework for easily rate controlling, instead of making individual TCP connections with the neighbors, a unified packet sender is used. In transmission, flow to neighbor i will report a rate for sending packets, say r_i. The sender will use a aggregated sending rate of $\sum r_i$ instead of individual sending rate r_i. Flow i can be controlled using a weight w_i, where $w_i = r_i / \sum r_i$. A round robin scheduler will be used to distribute packets proportional to fit the rate. In Fig. 3.6, suppose the rate with AIMD control for the four destinations are 400Kbps, 300Kbps, 200Kbps and 100Kbps, respectively. The rate sending out the packets for the host will be 1Mbps. In average, 4 out of 10 packets will be sent to neighbor 1, 3 packets will send to neighbor 2, and so on and so forth.

Packets sending out from one host are controlled using unified packet sender. This enables the smoothest traffic. The intervals of packets for every router in the path will be approximately equivalent if no congestion is encountered.

Each host in the overlay will use that algorithm to control the packets. The intermediate host in a multiple path transfer will do some additional tasks. The intermediate hosts will buffer the data from sending hosts in order to make delivery consistent. The intermediate host will feedback the delivery rate to the sender so that the sender can adjust the sending rate. The receiver should have a buffer for rearranging packets that are out of order.

Using MultiServ in ISP level, bandwidth may be reserved for the aggregate flow to achieve better throughputs. More aggressive protocols can also be used to enhance performance. For example, the edge server may set up a constant total bit rate for multiple paths, and then distribute the packets to each path by balancing the loss rate among all the paths.

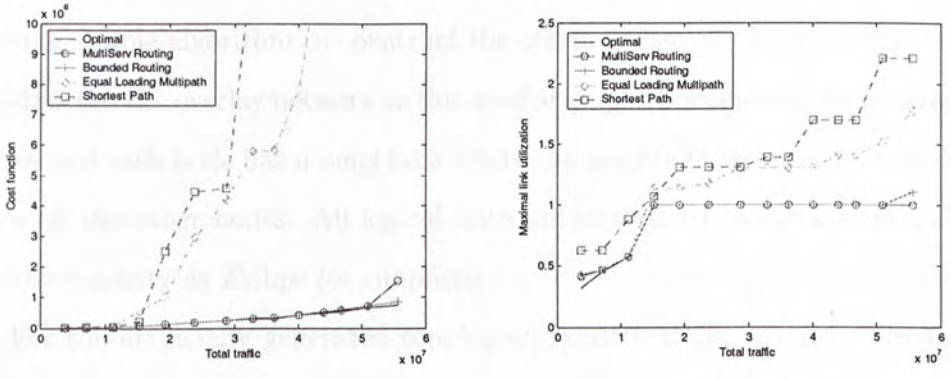
The benefits of the joint congestion control are of three folds. (a) The burst of traffic will be smoothed. The unified and rate-based sender sends data in a smooth way which will decrease the burst of traffic. (b) Rate control scheme can be easily applied. To adjust the rate for each path, users just need to modify their sending rate and the weight. (c) The aggregate flow has more control in QoS. Flows are aggregated so the hosts can easily adjust the rate for each flow to enable rich flow control.

3.4 Performance Evaluation

In order to evaluate the effectiveness of our approach, we conducte both simulation and experiments in real overlay network. The purpose of simulation is to compare different routing strategies. The purpose of real experiments is to explore the efficiency of overlay construction methods and compare potential performance of MultiServ data forwarding with IP network data forwarding.

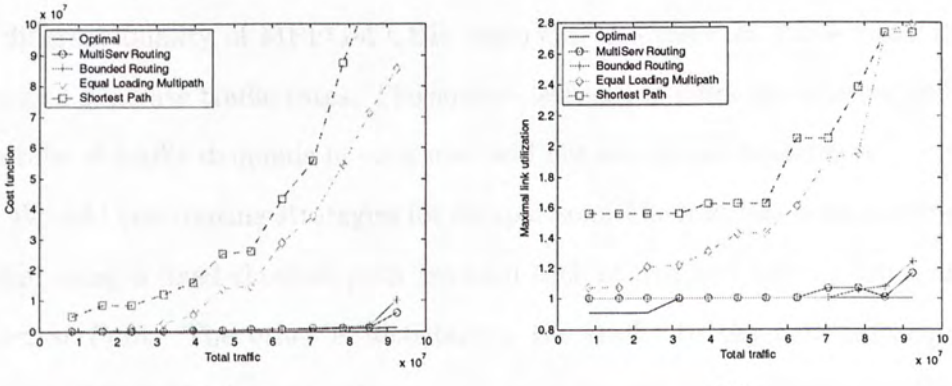
3.4.1 End-to-end streaming

Since the overlay network is constructed by special algorithm, the topology of overlay network typically is different from the topology in IP network. The nodes in the overlay network typically select neighbors from random nodes. Therefore



(a) (b)

Figure 3.7: Cost function and maximal link utilization vs total traffic in a 50 node 200 edge graph



(a) (b)

Figure 3.8: Cost function and maximal link utilization vs total traffic in a 100 node 400 edge graph

in large scale overlay network the topology is likely to be formed as random graph, where each potential link has almost same probability to be connected. In similar way, the neighbors could be considered as random nodes in the overlay network. This can be observed in popular peer-to-peer sharing network which is using simple algorithm to construct the overlay network. So in simulation we constructed the overlay network as this kind of graph. Specifically, we generate m nodes and each node has n neighbors where the neighbors are selected randomly from all the other nodes. All logical links are considered as symmetric and we set the capacity as 2Mbps for simplicity.

For the artificially generated topologies, random traffic matrices are generated. As an application layer protocol, we designed an end-to-end media streaming scenario in the simulation. Consider all users provide streaming service and streaming demands are generated from some of users. We consider the traffic as different quality of MPEG-4 CBR video clip and use $r = 300 + \text{rand}(1000)$ Kbps to generate traffic rates. The source-destination pairs are selected so that the rate of traffic demands in each user will not exceed its capacity.

We add two routing strategies for comparison. The first one is forwarding all traffic using a fixed shortest path between each source and destination, named *Shortest Path.* The other is distributing the traffic to the next hop equally among all possible shortest paths in each node, named *Equal Loading Multipath.* For example, if the node has two possible neighbors which can lead the traffic to the destination through shortest path, then both neighbors will deliver half of the traffic. In this method, the traffic can be divided multiple times in the intermediate nodes to distribute the traffic to as many links as possible. In our

experiments, the optimal routing is denoted as *Optimal*, the MultiServ routing strategy is denoted as *MultiServ Routing* and the MultiServ routing with bounded complexity is denoted as *Bounded MultiServ.*

Since the MultiServ Routing and Bounded MultiServ are performed in each node without knowing the whole traffic status in the overlay network, the experiments of these methods meet a little difficulty. In fact, the traffic input is necessary to be aware by these algorithms to generate the traffic output. However the traffic output is the traffic input of the neighbors, this will sequentially affect the status of forwarding. Therefore we need to simulate the real network as precise as possible which can reflect the phenomena. In our experiments we assume the delays of links are similar, thus the traffic can be delivered in steps. The first step the traffic flow out of the source and to its neighbor, named the first forwarder, the second step the traffic will flow to the first forwarder's neighbor, the second forwarder, and so on until the traffic reaches its destination. For each step the algorithm is running in each node according to the traffic input. We stop running the algorithm until the system enters a balanced status where the traffic flow in each node becomes stable and all the traffic reaches its destination.

We use a cost function described in [15] to compare performance of different strategies. The idea behind that is that it is cheap to send flow over a link with low utilization and expensive when the link is heavily loaded. It will be heavily penalized if the utilization is over 100%. Define c_{ij} and U_{ij} as the capacity and utilization of link (i, j):

$$\text{cost} = \sum_{(i,j)\in E} c_{ij} f(U_{ij}) \qquad \text{where}$$

$$f(x) = \begin{cases} x, & 0 \le x \le 1/3 \\ 3x - 2/3, & 1/3 \le x \le 2/3 \\ 10x - 16/3, & 2/3 \le x \le 9/10 \\ 70x - 178/3, & 9/10 \le x \le 1 \\ 500x - 1468/3, & 1 \le x \le 11/10 \\ 5000x - 16318/3, & 11/10 \le x \end{cases}$$

We use Matlab V6.5 to simulate the overlay network and solve the optimization problems. In Figure 3.7(a) and 3.7(b), we plot the cost and the maximal link utilization with different traffic demand for different routing strategies on a 50 node 200 edge graph. The traffic demand is from 5 to 70 streaming requests. Figure 3.8(a) and 3.8(b) plots the same content on a 100 node 400 edge graph with 10 to 120 streaming requests. It can be shown that when the traffic increases, the costs of Equal Loading Multipath and Shortest Path method increase sharply. While the MultiServ Routing and Bounded MultiServ method achieves similar cost comparing with Optimal method. On the maximal link utilization, the MultiServ Routing and Bounded MultiServ method is also similar with the Optimal method, while Equal Loading Multipath and Shortest Path method need more than two times of link capacity to deliver the traffic. From the result, the MultiServ Routing and Bounded MultiServ methods show near optimal performance under reasonable traffic load.

3.4.2 Application-layer multicast

In this simulation, application layer multicasting is performed using MultiServ Model. Imagine there are g multicast groups where each group i has one source with u_i users and the streaming rate is r Kbps. For simplicity, each group forms a binary multicast tree. Each newly joined user selects an existing user in the tree where the user has less than two children and the hop distance is minimal in order to generate less traffic in the overlay network.

In this simulation, large overlay network is formed in order to see the performance of different bandwidth strategies at large scale. The overlay network is still formed as random graph where each node has 4 neighbors and the capacities of all logical links are 1Mbps for simplicity. 20 multicast sources are randomly selected in the overlays where the rates are all 300Kbps, the traffic demands are constructed using the following ways: a random user is selected and it will pick a random multicast group which will add a traffic request for the overlay network.

In Figure 3.9, we plot the maximal link utilization with 50 to 500 multicast users for different routing strategies on a 500 node 2000 edge graph. Figure 3.10 plots the same content on a 1000 node 4000 edge graph with 100 to 1000 multicast users. In the figure we do not plot result of Optimal method because the global optimization needs to solve a linear programming problem which has millions of variables and constraints where we cannot afford the computation cost. For comparison our MultiServ Routing method only has around $m * n$ variables and similar number of constraints which are less than 10000 and still can be solved in normal PCs.

It can be seen from all the results that the MultiServ Routing and Bounded

MultiServ methods achieve similar performance in all simulation, which is far better than Equal Loading Multipath and Shortest Path methods. For example, from Figure 3.9, in 500 node 2000 edge graph with 1Mbps link capacity, more than 600 multicast users with 300Kbps rate can be supported using our strategies however using single shortest path only less than 150 users can be supported. From Figure 3.10, in 1000 node 4000 edge graph with 1Mbps link capacity, more than 1300 multicast users can be supported using our strategies and less than 400 users can be supported using single shortest path. It will be interested to consider the performance when the multicast content can be retrieved from multiple sources, but due to the limit of space it will not be discussed in this article.

The proposed routing strategies also show good scalability, stability and fast response. Illustrated in Figure 3.11, In 500 node overlay network with 500 multicast users, typically no more than 10 steps it can achieve a stable status where almost all traffic arrive its destination and the value of cost function have less than 1% difference between successive steps, which shows the utilization of links are quite stable.

The logical link capacities may change due to the congestion in underlying network. The performance of our routing strategies are also been evaluated under such situation. In 500 node overlay network with 500 multicast users, the link capacity are reset using $c = 750 + \text{rand}(500)$ Kbps every five steps. The cost function is illustrated in Figure 3.12. We can see our routing strategies can adapt the varied capacity and perform the data forwarding quite well, it is observed that no links are overloaded and all traffic can safely reach its destination.

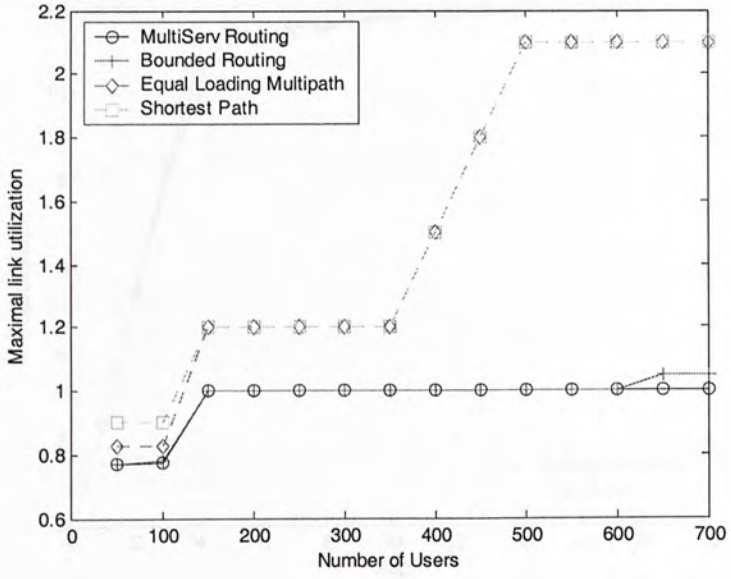


Figure 3.9: Maximal link utilization vs number of multicast users in a 500 node 2000 edge graph

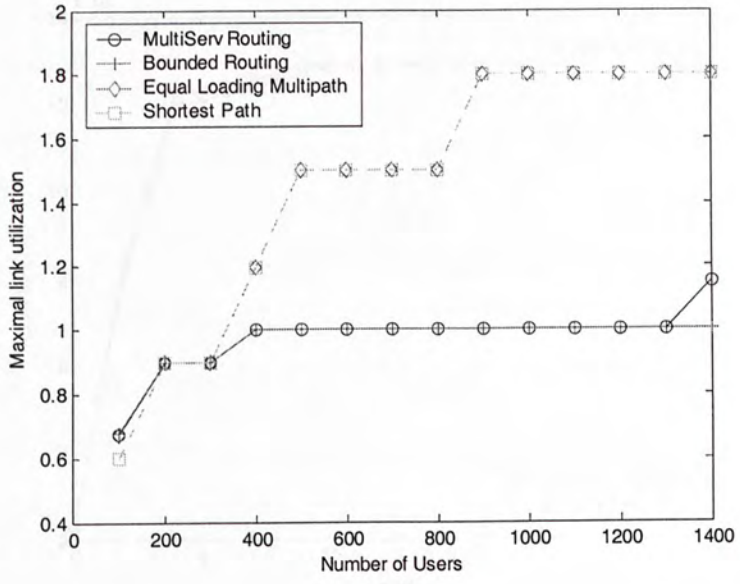


Figure 3.10: Maximal link utilization vs number of multicast users in a 1000 node 4000 edge graph

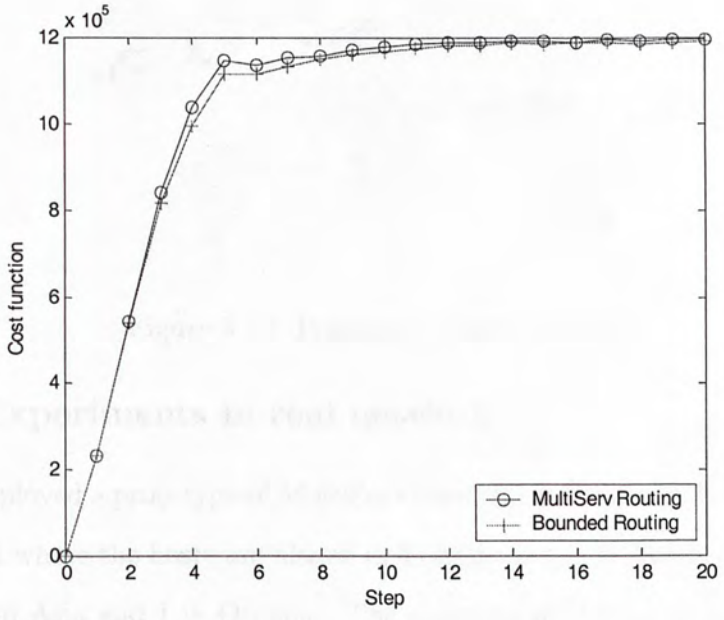


Figure 3.11: Cost function vs steps in a 500 node 2000 edge graph with 500 multicast users

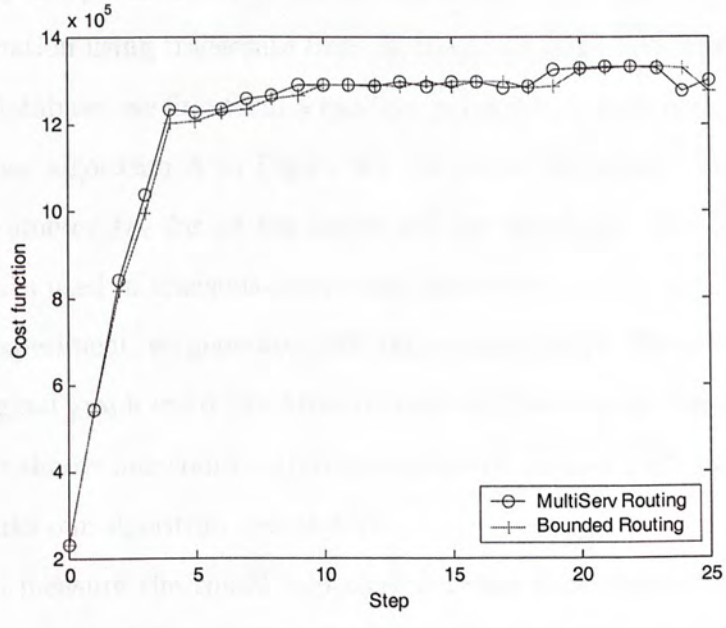


Figure 3.12: Cost function vs steps with varied link capacities in a 500 node 2000 edge graph with 500 multicast users

Figure 3.13: Positions of the 12 hosts

3.4.3 Experiments in real network

We have deployed a prototype of MultiServ platform in Planetlab hosts. 12 hosts are selected where the hosts are placed in 4 continents, 5 in North America, 4 in Europe, 2 in Asia and 1 in Oceania. The positions of the nodes are illustrated in Figure 3.13.

To verify the performance of our overlay construction algorithm, we obtain path information using traceroute from all the hosts and build a path database. Using the database, we first form a random graph which each node has 4 neighbors, and use algorithm A in Figure 3.2. to refine the graph. The average of disjoint parameter D_N for all the nodes will be calculated, this is the average common links used in transmission in each path pair.

In the experiment, we generate 1,000,000 random graph, the average common links of original graph are 6.16. After refinement, the average common links are 5.31, where almost one common link is eliminated in each path pair. The least common links our algorithm gets is 4.97.

We also measure the round trip time between these hosts, the round trip time is measured in different time of the day and the average is calculated. It is found that 61 of 132 round trip time could be decreased if the data is transferred

Table 3.2: Neighbor set of nodes in the overlay

Site	IP Address	Neighbors			
ucb	169.229.51.250	hp	uw	cam	cuhk
cuhk	137.189.97.17	ucb	tw	cam	tub
tw	140.109.17.180	cuhk	tor	mit	uts
hp	204.123.28.51	ucb	mit	diku	tor
uw	128.95.219.192	ucb	mit	tub	uts
mit	18.31.0.190	tw	hp	uw	uuse
tub	130.149.49.26	cuhk	uw	diku	cam
tor	128.100.241.67	tw	hp	diku	uts
uts	138.25.15.194	tw	uw	tor	uuse
cam	128.232.103.201	ucb	cuhk	tub	uuse
diku	192.38.109.143	hp	tub	tor	uuse
uuse	193.10.133.128	mit	uts	cam	diku

through other intermediate nodes to the destination. For example if we transfer data from ucb to tw through hp lab the round trip time could be decreased from 0.229s to 0.167s.

We pick a neighbor set which has 5.00 average common links in each path pair illustrated in Table 3.4.3. We use this neighbor set to form the overlay network for further experiments.

Lots of simulation of data transmission has been done in the situation with fixed bandwidth in each logical links. In real experiment, repeating the same experiment will result in similar result. Therefore we would like to explore the potential performance of the overlay network, that is, the maximum data transmission rate with the help of neighbors in the real experiment.

To achieve the purpose, first we have designed a program to estimate the throughput in a TCP transmission. The program launches a TCP transmission to transfer data without constraints between two hosts and last for 20 seconds. To eliminate the effect of TCP slow start, we use the average transmission rate

Table 3.3: Transmission time from ISP 1 to other ISPs.

KBps	ucb	cuhk	tw	hp	uw	mit	tub	tor	uts	cam	diku	uuse
ucb		518/210	723/145	442/227	2022/1254	791/378	716/175	577/544	824/185	794/231	800/165	831/172
cuhk	1007/323		1744/1190	373/32	908/354	824/266	673/194	675/226	733/164	769/248	660/229	871/225
tw	905/253	1668/1260		435/26	718/272	663/311	783/198	633/317	711/165	675/234	609/194	732/196
hp	1208/692	471/181	644/79		1283/163	505/128	421/260	423/168	579/72	805/25	520/45	308/70
uw	1796/1248	496/188	769/142	378/95		795/408	751/184	598/525	819/222	671/227	689/165	819/169
mit	727/376	467/138	619/149	419/75	1198/389		859/269	549/726	845/137	828/399	939/233	992/240
tub	535/156	397/117	513/95	325/20	589/206	729/258		556/209	651/96	1171/346	1083/133	1269/261
tor	1382/787	574/202	676/309	505/73	1756/796	1125/1256	1170/426		861/297	954/604	1152/385	1409/411
uts	794/316	435/164	689/165	463/23	1149/427	807/269	649/98	610/297		618/202	613/177	808/177
cam	528/217	369/125	585/123	440/27	991/228	874/409	1825/580	626/330	811/106		1399/639	1736/669
diku	588/247	444/173	634/195	423/28	1165/315	1007/465	2119/701	764/379	765/177	1973/1250		2403/1188
uuse	632/276	387/224	668/196	377/28	1135/325	976/490	2046/837	738/387	837/178	1975/1075	1873/1253	

in period of 1020 seconds as estimation of the rate of a TCP transmission.

Using the program, we estimate two kinds of transmission rates between two sites. One is the rate of direct connection and the other is the rate with the help of the neighbors, i.e., transmission to the destination through the neighbors. The second rate is considered as the potential transmission rate under MultiServ model.

The experiments were repeated 6 times in different time of a day and the average rates (KBps) of all source destination pairs are recorded in Table 3.3. A shaded cell indicates that the corresponding source destination pair has smaller round trip time using MultiServ model than direct connection. Table 3.3shows that the potential rate of transmission in MultiServ model is much higher than direct connection, especially in some sites with poor QoS to other sites, such as the hp site. This result suggests MultiServ a promising platform for heavy transmission tasks.

3.5 Summary and Future Work

This chapter proposes a new model called MultiServ to alleviate the congestion and to provide better quality of service for end-host using overlay network. A special overlay is proposed in this model. Meanwhile, distributed routing strategies are proposed to balance the utilization of different logical links and enhance the performance of data forwarding. The advantages of MultiServ are summarized as follows.

- Scalable and Stable: The overlay uses a heuristic construction algorithm which can perform in stable way and can be easily extended to large scale, and the routing strategies can also work well in large overlay network with varied link capacities;
- Higher performance, rich QoS control and enable more services: The aggregate flow has enabled rich control on flows, the model could enable many services which may not be practical or scalable in IP network, such as large scale streaming and multicasting;
- Easy to deploy: For ISPs, edge servers with MultiServ can balance traffic; for users, simple system service and MultiServ-aware software enables the service.

We have performed extensive simulation and real experiments to verify our model. From the result we can see the routing strategies perform pretty well under different applications in large scale overlays and the constructed overlay shows good potential performance in real network.

This is an ongoing work. We will implement the proposed model and deploy it to real environment for further evaluations. As shown in the simulation, application-layer multicast based on this model is also considered as a promising research direction.

Chapter 4

DDS: Distributed Dynamic Streaming

In this chapter a new framework called *dynamic distributed streaming* (DDS) is presented for both on-demand streaming and live-streaming using overlay network. A user model is built and the user data outage in streaming is derived and compared with application-layer multicast. Results confirmed by simulations show that DDS can perform much better in dynamic user environment. In large overlays, the data outage in DDS is as low as 10% of that in tree based application layer multicast in a highly dynamic user environment.

4.1 Motivation

Streaming bandwidth sensitive media from a single source to a large number of users is a difficult task on the Internet. We can divide streaming into two

categories: on-demand streaming and live-streaming. A possible way to serve on-demand streaming is the use of content distribution network (CDN) which can be considered as a distributed caching technique. Generally speaking, caching, including CDN, is a method of using storage to reduce transmission. However, media data are large objects that are difficult to store in the cache server. Further more, for live streaming, the storage cannot help much. In this case, multicast is a natural paradigm for distributing live media data. However IP multicast is not widely deployed. Therefore application-layer multicast is served as a alternative solution. Unlike IP multicast where data packets are replicated at routers inside the network, in application-layer multicast data packets are replicated at end hosts. The data delivery path in application-layer multicast is an overlay tree.

A key challenge in constructing a resilient application-layer multicast protocol is to provide fast data recovery when an overlay host failure breaks down the data delivery paths. Overlay hosts are processes on regular end-hosts which are potentially more susceptible to failures than the routers.

In this chapter, we present a new framework for both on-demand streaming and live-streaming in application layer called Distributed Dynamic Streaming (DDS). Simple examples of application layer multicast and DDS are illustrated in Fig. 4.1. Similar to application-layer multicast, DDS builds an overlay network and the content is replicated at end-hosts. Different from the traditional streaming model, in DDS, the overlay network is a random graph topology and streaming content will not come from one single upstream source, the content will be delivered like gossiping, from all available paths to the clients. Several application layer streaming or content distribution solutions were presented [16, 18], these work are based on tree structure where each host has only one upstream

provider. Cooperative streaming [19], flash crowds handle [22] and multicast data recovery [17] are methods for improving the performance of streaming in application layer. The main focus is on resilient and efficient streaming under a dynamic user environment. The user model is built and the user data outage in streaming is derived and compared with application-layer multicast. Results confirmed by simulations show that DDS can perform much better in dynamic user environment.

The remainder of the chapter is organized as follows. In section 2 we introduce the DDS protocol. Section 3 contains the analysis the streaming data outage in tree and graph model given user model. the model is verified using simulation in section 4. In section 5, we summarize the approach and propose the future work.

4.2 Distributed Dynamic Streaming

The principle of distributed gossip communication is simple. In each time step, each host v in the network selects some random host w as a communication partner and exchange information with each other, therefore over a period of time, information spreads through the network in an "epidemic fashion".

Quality of streaming for each host is the primary design objective of the DDS protocol. The design can be divided into two parts: algorithm to construct an overlay and protocol to deliver streaming content using gossip based techniques among users.

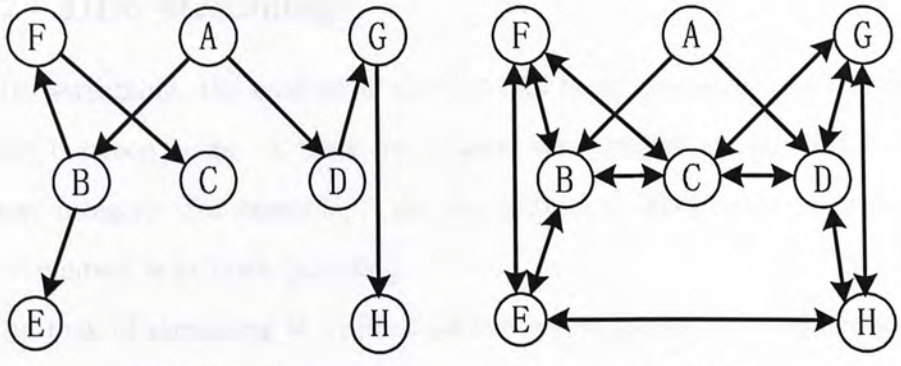
4.2.1 DDS overlay construction

DDS overlay can be considered as an unstructured overlay network. The unstructured overlay network can be viewed as an undirected graph, where nodes correspond to hosts and edges correspond to open connections maintained between the hosts. Two connected hosts are known as *neighbors*. Several large unstructured overlay networks are running on the Internet. One of the earliest unstructured overlay network uses the Gnutella protocol [25]. The construction protocol of Gnutella is simple, robust and scalable for millions of users to participate. Information searching is normally hard to be scalable in large unstructured network. But for streaming, searching is not necessary and so not a problem.

Here we briefly describe the construction of an unstructured overlay. When a client wishes to join an overlay network, it first contacts a bootstrap server to obtain a preliminary list of neighboring candidates identified by a list "IP:Port" combinations. The client then contacts these candidates to find satisfactory neighbors. Depending on the satisfaction of certain criteria, such as limit of neighbor size, the neighbor and the client can then form a neighbor relationship. The overlay will grows when neighboring hosts are joined following the above process.

In DDS design, the neighbor size n of each client should be carefully chosen. If n is too small, gossips cannot be passed effectively. On the other hand, if n is too large, a host may be overloaded by gossips. In Gnutella system, the neighbor size setting is typically between 3 and 4. Our design adopts this setting.

To improve the streaming quality, the constructed overlay should be locality aware, for example, the overlay host chooses neighbors with small delay and large



(a) Multicast (b) GBS

Figure 4.1: A comparison of Multicast and DDS

bandwidth. Several techniques [20, 23] were proposed to construct a locality aware overlay. These can also be directly adopted in construction of the DDS overlay.

To improve the overlay performance for dynamic user environment, similar with [17], each host in the overlay will maintain a list of good neighbor candidates, thus, when one neighbor leaves the overlay, the host can directly negotiate with the best neighboring candidate to form a new neighbor relationship. This practice can restore the overlay as soon as possible under dynamic user environment.

In an overlay with n hosts, the hop radius (The average logical hops between two hosts) is in the order of $\log(n)$ [24]. On the other hand, a Gnutella overlay network can remain connected even if 30% of the randomly selected hosts are removed [21]. Those characteristics suggest a stable platform for streaming tasks.

4.2.2 DDS streaming

In DDS streaming, the content is divided into basic elements called *bricks* to transfer between hosts. A brick has a fixed size. Bricks are labelled by non-negative integers. For example, if the size of brick is 4096 bytes, then the nth byte of content is in brick $\lfloor n/4096 \rfloor$.

The task of streaming is to fetch all the bricks timely and efficiently. The protocol is illustrated in Fig. 4.2. The DDS system is composed of three parts: buffer manager, scheduler and sender.

1. **Buffer Manager:** buffer manager is responsible for managing buffer, receiving data from scheduler and feeding data to the sender. In DDS we may need to maintain a large buffer for providing contents to other hosts.

2. **Scheduler:** Scheduler is a key component in the system. It is responsible to decide at a specific moment which brick should be fetched from where. We will explain the details later.

3. **Sender:** The task of sender is rather simple, upon receiving a request from neighbor for content, it will get the content from the local buffer and send it out to the neighbors.

The procedure of delivery begins from establishing the connection to the neighbor. Once connected, the host will fetch a buffer map from the neighbor, where they know which brick is available in the neighbor. Then the transmission begins. In streaming, a buffer map change should notify all the neighbors. For example, if a brick is received, the host will send a message to all the neighbors saying that it has the brick, the message should include the brick sequence

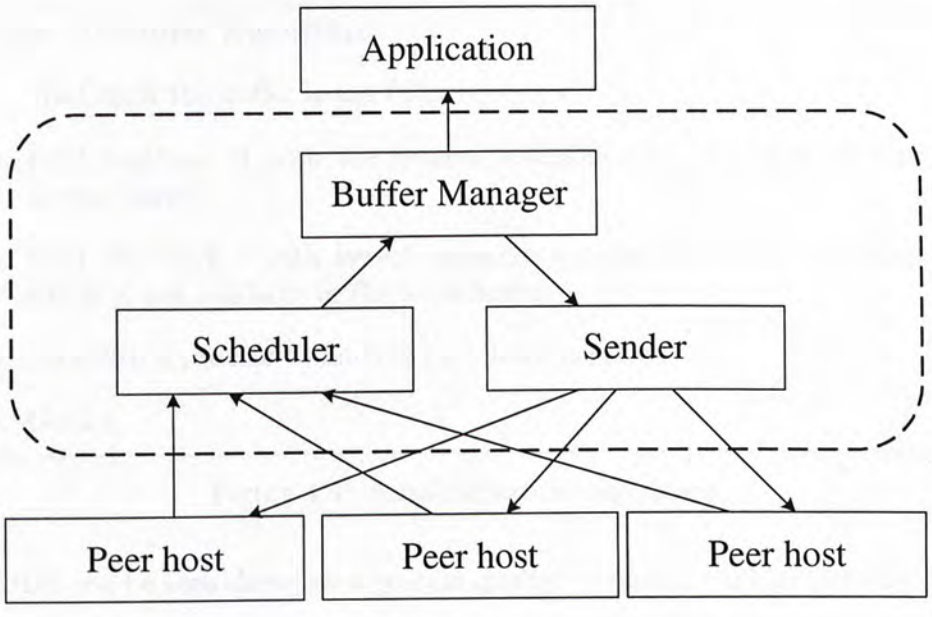


Figure 4.2: DDS delivery structure illustration.

number with a flag indicate that the brick is received. The message can be as small as 5 bytes. This will ensure that the host have the buffer status of all the neighbors at any time.

The scheduler will make a decision of transmission given the buffer status of all neighbors. A simple scheduler algorithm is given in Fig. 4.3, where available needed brick with lowest sequence number is transferred from each neighbor in turn. This enable balanced data delivery from each neighbor.

There might be other schedulers which can provide better delivery performance for the system. For example if the delivered sequence of bricks is randomized, the availability of content in the system may be larger. This might be an interesting research topic and is out of scope of this thesis.

Simple Scheduler Algorithm

1. Wait until the buffer is not full;
2. Find neighbor A with the longest available time (no data fetching for longest time);
3. Find the brick s with lowest sequence number available in neighbor A which is not available in the local buffer;
4. Establish a process to fetch brick s from neighbor A;
5. Goto 1

Figure 4.3: Simple scheduler algorithm.

DDS can be considered as a general quality of service content delivery platform and thus we want the delivered content to be precise and complete. Therefore in the design of protocol we do not focus on incorporating complicated streaming techniques such as FGS (Fine Granularity Scalability) streaming or MDC (Multiple Description Coding). However our protocol can easily be extended for use with these techniques.

4.3 Performance Analysis in Dynamic User Environment

In this section, we analyze the data outage in application layer multicast and in DDS.

4.3.1 Basic definition and user model

Consider an overlay network with n nodes, the overlay here can be any topology. For example, in scalable application layer multicast protocol NICE [16], the overlay network is a tree with up to m children in each non-leaf node. In DDS, the overlay is considered as a near random graph. For simplification of analysis, neighbors can be considered similar with children in the tree structure, so each node in DDS also has m neighbors. We assume the streaming source (the root in the multicast tree) is stable thus the streaming source will be considered as a special node which will not fail in the following discussion.

In the overlay network, each node represents a user who can leave at any time, the departure of a user can be considered as the failure of a node. Therefore the failures of nodes are independent events. Ideally, if the connection time any node in the overlay is exponential distributed with identical parameter λ, then the failure can be modelled as a Poisson process, which has the following properties: At any particular time period $[t, t+\epsilon]$, the probability of the failure of any node is $1 - e^{-\lambda\epsilon}$. In general, the node may have arbitrary distribution in online time and arrival time, but as long as we treat the node identical and independent, it is safe to say that in any small enough time period $[t, t+\epsilon]$, the probability of the failure of any node is approximately identical, which can be represented by a function of t and ϵ, say $f(t, \epsilon)$. Therefore we can define the failure rate for any node as $R(t)$, where $R(t) = \lim_{\Delta \to 0} f(t, \Delta)/\epsilon$. We assume our analysis period $[t, t+\epsilon]$ is stable, that is $R = R(t)$ is a constant in the period. Therefore we can have the following estimations: The average failure of the overlay network is $nR\epsilon$ and the probability of failure of any host in the period is $R\epsilon$.

A node failure in the overlay may cause data outage in direct downstream

nodes until the time the data delivery tree is reconstructed. Consider a node has a buffer of time t_b in receiving streaming content, therefore if the downstream node cannot find another node to fetch stream (that is a new neighbor in DDS and parent in the tree structure) after t_b time from the failure, an outage of data may occur. This can be considered as identical event for each host, thus we can define that each direct downstream of the failure node has equal probability P_o of not finding a new upstream node within t_b time, which will be useful in our following analysis

4.3.2 Data outage in tree topology

In the tree topology, each node has up to m children. To simplify the estimation, we assume the tree is complete, i.e., at most one non-leaf node has less than m children. Therefore the height of the tree is $h = \lfloor \log_m((m-1)n+1) \rfloor - 1$.

For node X in the tree, any ancestor failure may cause the data outage if the downstream node can not find a new parent in time. For example, in Fig. 4.1(a), if node B fails, the data outage probability of node F and E is P_o. If data outage occurs in node F, then it will also occur in node C since it cannot receive the data where F do not receive[1]. For example, if an ancestor fails, the probability of data outage is P_o; if two ancestors fail, the probability of data outage is $P_o + (1 - P_o) * P_o$. So let $\mathcal{A}(u, i)$ be the event that node u has data outage with i ancestor failures, the probability of the event is

$$\mathrm{P}[\mathcal{A}(u,i)] = 1 - (1 - P_o)^i$$

[1]There might be other data recovery techniques which may decrease the data outage probability, but for simplicity of analysis, we do not address the techniques here and use the basic model.

Let $\mathcal{B}(k,i)$ be the event that i nodes fail in k ancestors in period $[t, t+\epsilon]$, then probability of the event is

$$\mathrm{P}[\mathcal{B}(k,i)] = \binom{k}{i}(R\epsilon)^i(1-R\epsilon)^{k-i}$$

Let $\mathcal{C}(u,k)$ be the event that node u in depth k has data outage in period $[t, t+\epsilon]$. Then the probability of the event is

$$\begin{aligned}
& \mathrm{P}[\mathcal{C}(u,k)] \\
= & \sum_{i=0}^{k-1} \mathrm{P}[\mathcal{B}(k-1,i)] \cdot \mathrm{P}[\mathcal{A}(u,i)] \\
= & \sum_{i=0}^{k-1} \binom{k-1}{i}(R\epsilon)^i(1-R\epsilon)^{k-1-i} \cdot (1-(1-P_o)^i) \\
= & \sum_{i=0}^{k-1} \binom{k-1}{i}(R\epsilon)^i(1-R\epsilon)^{k-1-i} - \\
& \sum_{i=0}^{k-1} \binom{k-1}{i}((R\epsilon)(1-P_o))^i(1-R\epsilon)^{k-1-i} \\
= & 1-(1-P_oR\epsilon)^{k-1}
\end{aligned}$$

The average data outage in the tree $\mathrm{E}(D_t)$ can be calculated by adding probability of all the nodes,

$$\begin{aligned}
\mathrm{E}(D_t) &= \sum_{i=2}^{h-1} m^i * \mathrm{P}(\mathcal{C}(u,i)) + (n - \frac{m^h-1}{m-1}) * \mathrm{P}(\mathcal{C}(u,h)) \\
&= \frac{m^h-m^2}{m-1} + (n-\frac{m^h-1}{m-1})(1-(1-P_oR\epsilon)^{h-1}) \\
&\quad -m^2(1-P_oR\epsilon)\frac{(m(1-P_oR\epsilon))^{h-2}-1}{m(1-P_oR\epsilon)-1} \qquad (4.1)
\end{aligned}$$

From Eq. (4.1) we can estimate the average node data outage in the tree structure given n, m, P_o and $R\epsilon$. For example, given $n = 5000, m = 3, P_o = 0.2, R = 0.001$ and $\epsilon = 10$, the estimation of $\mathrm{E}(D_g)$ is 56.46. That indicates the following facts: In a 5000 nodes multicast tree with 3 children in each node, assume each

node has 20% possibility to be short of data when its parent fails, if in a future 10 seconds period, 50 nodes fail while new nodes replace the current nodes and the tree keeps the original size. then on average 56.46 data outage will occur in the period.

Note the discussion above is the best condition in such a tree structure. In real environment, the performance may be worse. For example, if the tree is not a complete tree, the height of the tree will increase so that the probability of data outage will also increase since nodes may have more ancestors and thus more probability on the failure of the ancestors. Also, if is in the tree maintenance protocol, low depth nodes will process more messages and thus it might be more vulnerable. These nodes therefore have higher probability of failure. This will also increase the data outage percentage since it has more descendants and affect more nodes when departure. So in practical environment, (4.1) is a conservative esitmate.

4.3.3 Data outage in DDS

In DDS each node has m neighbors where all the neighbors can provide streaming content to the node, the facts lead to a direct implication that the delivery of streaming content can survive in worse network conditions and user transience than the tree topology where only one data supplier is available for one node. For example, the protocol of DDS can greedily seeks all available parts of the streaming content from the neighbors, therefore even all neighbors have data outage, the node can still survive in very high probability. So we will omit the situation and focus on the event of a direct neighbor failure.

Failure of a node may cause the data outage of all its neighbors. Similar

with tree topology analysis, here we know that the probability is P_o when the downstream node cannot find another node for stream delivery after t_b time from failure of neighbors. However when it cannot find a new node in given time, the data outage still may not occur. For example, in Fig. 4.1(b), consider a specific failure at node B, all the neighbors of B may be affected by the failure. Unlike tree topology, for the node F who has lost the neighbor B, F still can continue the streaming from the content node C and E provides. Generally speaking, suppose X fails and X is Y's neighbor, if one or more neighbors of the host Y can delivery the part which X should originally deliver to Y in the time period node Y tries to find a new neighbor, the data outage would not occur. This process can greatly decrease the probability of data outage.

In the following analysis, a parameter P_s is defined as the probability for any node to serve full streaming service during the time its neighbor needs to find a new neighbor. Recall in the multicast tree structure almost all non-leaf nodes need to deliver m copies of stream, we believe serving full streaming service for the node at a specific time period should not be a hard task thus P_s is reasonable to be a large value, for example we can assume $P_s > 0.5$.

Therefore for node A, the data outage only occurs when in ϵ time, the node cannot find a neighbor and none of the remaining neighbors are capable to serve the streaming. We define $\mathcal{O}(A)$ be the event node A has data outage, So the probability the event is:

$$\begin{aligned} & \mathrm{P}[\mathcal{O}(A)] \\ = \; & P_o \sum_{i=1}^{m} \mathrm{P}(i \text{ neighbors fail}) \cdot \\ & \mathrm{P}(\text{none of remaining neighbors can serve streaming}) \end{aligned}$$

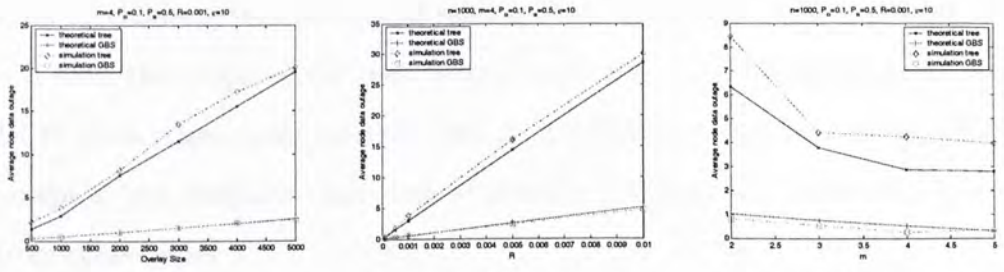
$$
\begin{aligned}
&= P_o \sum_{i=1}^{m} \binom{m}{i} (1 - R\epsilon)^{m-i} (R\epsilon)^i \cdot (1 - P_s)^{m-i} \\
&= P_o (\sum_{i=0}^{m} \binom{m}{i} ((1 - R\epsilon)(1 - P_s))^{m-i} (R\epsilon)^i \\
&\quad -((1 - P_s)(1 - R\epsilon))^m) \\
&= P_o((1 - P_s + P_s R\epsilon)^m - ((1 - P_s)(1 - R\epsilon))^m)
\end{aligned}
$$

Therefore the average data outage in the graph $\mathrm{E}(D_g)$ can be calculated by adding probability of all the hosts,

$$
\begin{aligned}
\mathrm{E}(D_g) &= nP_o((1 - P_s + P_s R\epsilon)^m - \\
&\qquad ((1 - P_s)(1 - R\epsilon))^m) \qquad (4.2)
\end{aligned}
$$

From Eq. (4.2) we can estimate the average outage number of nodes in the graph structure given n, m, P_s, t and $R\epsilon$. For example, given similar parameters with tree topology which is $n = 5000, m = 3, P_o = 0.2, P_s = 0.6, R = 0.001$ and $\epsilon = 10$, the estimation of $\mathrm{E}(D_g)$ is 4.82. This means that in a 5000 nodes DDS topology with 3 neighbors in each node, assume each node can serve full streaming service during the time its neighbor need to find a new neighbor in probability of 0.6. If in a future 10 seconds period, 50 nodes fail while new nodes replace the current nodes and the tree keeps the size around 5000. then on average 4.82 data outage will occur in the period, which is much smaller than in tree structure.

The result of Eq. (4.2) is a rough estimation. Since if not all neighbors fail, the streaming content can at least be partially provided and thus the time of buffer underflow will be delayed so that the streaming should survive more time. Further more, if no single neighbor can supply the full streaming, the collaboration of neighbors may provide full streaming content. This is not included in



(a) size (b) R (c) m

Figure 4.4: Data outage of overlay network with different metrics

estimation. So in practical environment, the Eq. (4.2) data outage might be of overestimation on data outage in the overlay.

4.4 Performance Evaluation

In this section we present a performance evaluation of both tree structure application layer multicast and DDS based on various of dynamic user environment.

4.4.1 Simulation setup

In all these simulations we model the scenario of a source node distributing streaming media to a set of nodes. We use a tree topology and random graph topology to build the overlay. Because the main objective of the simulation is to verify the analysis model in the previous section, in the simulation, we do not consider the bandwidth and link delay in the overlay network, and focus on the data outage in user dynamic environments.

The tree structure is constructed in the following way: When a new node joins the overlay, it will find the oldest node which can accept a new child (less

than m children) and get the streaming content from the node. This process will limit the height of the tree. Similar with tree topology, in construction of DDS, when a new node joins the overlay, it will find m random node which can accept a new neighbor (less than m neighbor) and get the streaming content from these nodes.

In both topology, end-hosts first continuously join and leave the multicast group. The join and leave rate for members are chosen to be equal (R) so that the average size of the group remained nearly constant (n).

The parameters in the simulation are chosen in the following ranges: n from 500 to 5000, m from 2 to 5, R from 0.0001 to 0.01, $P_o = 0.1$, $P_s = 0.5$ and we fix ϵ to 10 to see the short period phenomena of the overlay network, the short period can also reflect the long period as long as the overlay network is stably evolving.

4.4.2 Simulation results

In Fig. 4.4(a) the node data outage of overlay network with different sizes are shown. The size is from 500 to 5000 with $m = 4$ and $R = 0.001$. We show the theoretical and simulation result. In all the simulations, we run it for 100 times in each scheme and get an average to compare with theoretical result. It is shown that when the overlay becomes large, the data outage in tree structure increase significantly while data outage in DDS increases much slower. For example, for overlay large than 1000 nodes, the data outage in the graph is as low as 10% of that in the tree model. This indicate that DDS is more scalable in dynamic environment than tree based structure.

In Fig. 4.4(b) the node data outage of overlay network with different failure

rate are shown. R is from 0.0001 to 0.01 in a 1000 node overlay with $m = 4$. It is shown that when failure rate R of nodes increases, the data outage in tree structure increase significantly while data outage in DDS increases much slower. This indicates that DDS is more robust under high user transience.

In Fig. 4.4(c) the node data outage of overlay network with different neighbor number are shown. m is from 2 to 5 in a 1000 node overlay. It is shown that when neighbor number m increases, the data outage of both topology decreases, DDS shows much advantage over tree structure in all cases. In tree structure the decrease is more significant however increasing m in the tree indicates each non-leaf nodes should support more clients which may made them heavy loading. DDS, for comparing, the load of each node may increase when they have more neighbors but it will not be much since the load will be nearly equally distributed to all neighbors.

4.5 Summary and Future Work

In this chapter, we propose a new framework called dynamic distributed streaming for both on-demand streaming and live-streaming in application layer. User model is built and the user data outage in streaming is derived and compared with application-layer multicast. Results confirmed by simulations show that DDS can perform much better in dynamic user environment.

This is an ongoing work. We will implement the proposed framework and deploy it to real environment for further evaluations.

Chapter 5

Concluding Remarks

This thesis attempts to explore the problems in the area of multiple point communications. As we mentioned previously, there are three key issues: performance, scalability and stability. Our approach is working towards these points. The MultiServ architecture aims at improving the performance and scalability of data transmissions in overlay network. On the other hand, the DDS scheme addresses the problem of performance and stability in previous approaches. Simulations and experiments show significant advantages in our approaches compared with others.

In area of multiple point communication, undoubtedly a lot of work is yet to be done. This thesis just tried to tackle a little from the huge mountain, much is left to be dig later, in which I expect to find treasures.

Bibliography

[1] C. Fraleigh, S. Moon, C. Diot, B. Lyles, and F. Tobagi, "Packet-Level Traffic Measurements from a Tier-1 IP Backbone", Sprint ATL Technical Report TR01-ATL-110101, November 2001, CA.

[2] S. Iyer, S. Bhattacharyya, N. Taft, C. Diot, "An approach to alleviate link overload as observed on an IP backbone", in *Proc. of IEEE INFOCOM* 2003.

[3] D. Katabi, M. Handley, and C. Rohrs, "Congestion Control for High Bandwidth-Delay Product Networks", in *Proc. ACM SIGCOMM 2002*, August 2002.

[4] S. Floyd, "HighSpeed TCP for Large Congestion Windows", Internet draft, draft-floyd-tcp-highspeed-01.txt, work in progress, 2002.

[5] D. Wetherall, R. Mahajan, and T. Anderson,. "Under-standing BGP misconfigurations", in *Proc. ACM SIGCOMM 2002*, August 2002.

[6] Z. Morley Mao, R. Govindan, G. Varghese, and R. Katz, "Route Flap Damping Exacerbates Internet Routing Convergence", in *Proc. ACM SIGCOMM 2002*, August 2002.

[7] D. Andersen, H. Balakrishnan, M. Kaashoek, and R. Morris. "Resilient Overlay Networks", in *Proc. ACM SOSP*, 2001.

[8] L. Subramanian, I. Stoica, H. Balakrishnan and R. Katz, "OverQoS: Offering QoS using Overlays", in *1st Workshop on Hop Topics in Networks (HotNets-I)*, 2002.

[9] Z .Duan, Z. Zhang, Y. T. Hou, "Service Overlay Networks: SLA, QoS and Bandwidth Provisioning", in *Proc. ICNP'02.*

[10] S. Banerjee, B. Bhattacharjee, and C. Kommareddy, "Scalable application layer multicast", in *Proc. ACM SIGCOMM 2002*, August 2002.

[11] K. P. Gummadi, S. Saroiu, and S. D. Gribble, "King: Estimating Latency between Arbitrary Internet End Hosts", in *Proceedings of the 2nd Internet Measure-ment Workshop*, Marseille, France, November 2002.

[12] H. Balakrishnan, H. Rahul, and S. Seshan, "An Integrated Congestion Management Architecture for Internet Hosts", in *Proc. ACM SIGCOMM 1999*, September 1999.

[13] Y. Wang, Z. Wang, and L. Zhang, "Internet Traffic Engineering without Full Mesh Overlaying", in *Proc. IEEE INFOCOM 2001.*

[14] A. Sridharan, R. Guerin, C. Diot, "Achieving Near-Optimal Traffic Engineering Solutions for Current OSPF/IS-IS Networks", in *Proc. of IEEE INFOCOM 2003.*

[15] B. Fortz and M. Thorup, "Internet traffic engineering by optimizing OSPF weights", in *Proc. of IEEE INFOCOM* 2000.

[16] S. Banerjee, B. Bhattacharjee, and C. Kommareddy, "Scalable Application Layer Multicast," In *Proceedings of ACM SIGCOMM*, Pittsburgh, PA, August 2002

[17] S. Banerjee, S. Lee, B. Bhattacharjee, and A. Srinivasan, "Resilient Multicast using Overlays," In *Proceedings of ACM SIGMETRICS*, 2003

[18] Y. Chu, S.G. Rao, and H. Zhang, "A Case for End System Multicast," in *ACM Sigmetrics 2000*, Santa Clara, California, USA, 2000.

[19] V. N. Padmanabhan, H. J. Wang, P. A. Chou, and K. Sripanidkulchai. "Distributing Streaming Media Content Using Cooperative Networking," In *Proc. NOSSDAV*, May 2002.

[20] S. Ratnasamy, M. Handley, R. Karp, S. Shenker, "Topologically-Aware Overlay Construction and Server Selection". In *Proc. of Infocom 2002*

[21] S. Saroiu, P. K. Gummadi and S. D. Gribble, "A Measurement Study of Peer-to-Peer File Sharing Systems," in *Proc. of the Multimedia Computing and Networking (MMCN), 2002.*

[22] A. Stavrou, D. Rubenstein, S. Sahu, "A Lightweight, Robust P2P System to Handle Flash Crowds," In *Proceedings of IEEE ICNP 2002*, Paris, France, November, 2002.

[23] X. Zhang, Q. Zhang, Z. Zhang, G. Song, W. Zhu, "A Construction of Locality-Aware Overlay Network: mOverlay and its performance". To appear in *IEEE JSAC Special Issue on Recent Advances on Service Overlay Networks*

[24] X. Zhang, G. Song, Q. Zhang, and W. Zhu, "Performance Analysis in Unstructured Overlays", in *IEEE ICC'03.*

[25] Gnutella, http://www.gnutella.com

[26] napster, http://www.napster.com

[27] Kazaa, http://www.kazaa.com/

[28] edonkey, http://www.edonkey2000.com/

[29] bittorrent, http://bitconjurer.org/BitTorrent/

[30] The Global Grid Forum website: http://www.gridforum.org.

[31] D. Awduche et al., "Requirements for Traffic Engineering Over MPLS", RFC 2702, Sept. 1999.

[32] D. Awduche et al., "RSVP-TE: Extensions to RSVP for LSP Tunnels," RFC 3209, Dec. 2001.

[33] D. Awduche, A. Hannan, and X. Xiao, "Applicability Statement for Extensions to RSVP for LSP-Tunnels," RFC 3210, Dec 2001.

[34] R. Callon et al., "A Framework for Multiprotocol Label Switching," IETF Internet draft, work in progress, Nov. 1997.

[35] B. Jamoussi et al., "Constraint-Based LSP Setup Using LDP," RFC 3212, Jan. 2002

[36] T. Li, G. Swallow, and D. Awduche, "IGP Requirements for Traffic Engineering with MPLS," IETF Internet draft, work in progress, Feb. 1999.

[37] E. Rosen, A. Viswanathan, and R. Callon, "Multiprotocol Label Switching Architecture," RFC 3031, Jan 2001.

[38] M. A. Rodrigues and K. G. Ramakrishnan, "Optimal Routing in Shortest Path Networks," ITS'94, Rio de Janeiro, Brazil.

[39] Y. Rekhter and T. Li, "A Border Gateway Protocol 4 (BGP-4)," RFC 1771, March 1995.

[40] J. Moy, "OSPF Version 2," STD 54, RFC 2328, July 1997.

[41] "Intermediate System to Intermediate System Intra-Domain Routing Exchange Protocol", ANSI X3S3.3/87-150R, 1987-10-29.

[42] M. Castro, P. Druschel, A-M. Kermarrec, and A. Rowstron. "SCRIBE: A large-scale and decentralized application-level multicast infrastructure." in *IEEE Journal on Selected Areas in communications (JSAC)*, 2002.

[43] Y.-H. Chu, S. G. Rao, S. Seshan, and H. Zhang. "Enabling Conferencing Applications on the Internet using an Overlay Multicast Architecture." In *Proceedings of ACM SIGCOMM*, August 2001

[44] S. Deering and D. Cheriton. "Multicast Routing in Datagram Internetworks and Extended LANs." In *ACM Transactions on Computer Systems*, May 1990.

[45] P. Francis. "Yoid: Extending the Multicast Internet Architecture," 1999. White paper http://www.aciri.org/yoid/.

[46] J. Jannotti, D. Gifford, K. Johnson, M. Kaashoek, and J. O'Toole. "Overcast: Reliable Multicasting with an Overlay Network." In *Proceedings of the*

4th Symposium on Operating Systems Design and Implementation, October 2000.

[47] J.C. Lin and S. Paul. "RMTP: a reliable multicast transport protocol." In *Proceedings of IEEE Infocom*, March 1996.

[48] D. Pendarakis, S. Shi, D. Verma, and M. Waldvogel. "ALMI: An Application LevelMulticast Infrastructure." In *Proceedingsof 3rd Usenix Symposium on Internet Technologies & Systems*, March 2001.

[49] I. Stoica, R. Morris, D. Karger, M. F. Kaashoek, and H. Balakrishnan, "Chord: A scalable peer-to-peer lookup service for internet applications," In *Proc. ACM SIGCOMM*, August 2001.

[50] S. Ratnasamy, P. Francis, M. Handley, R. Karp, and S. Shenker. "A scalable content-addressable network." In *Proceedings of ACM Sigcomm*, August 2001.

[51] S. Ratnasamy, M. Handley, R. Karp, and S. Shenker. "Applicationlevel multicast using content-addressable networks." In *Proceedings of 3rd International Workshop on Networked Group Communication*, November 2001.

[52] I. Rhee, N. Ballaguru, and G.N. Rouskas. "MTCP: Scalable TCPlike congestion control for reliable multicast." In *Proceedings of ACM Sigcomm*, August 1998.

[53] L. Rizzo. "pgmcc: a TCP-friendly single-ratemulticast congestion control scheme." In *Proceedings of ACM Sigcomm*, August 2000.

[54] A. Rowstron and P. Druschel. "Pastry: Scalable, distributed object location and routing for large-scale peer-to-peer systems." In *IFIP/ACM International Conference on Distributed Systems Platforms (Middleware)*, November 2001.

[55] B. Zhang, S. Jamin, and L. Zhang. "Host multicast: A framework for delivering multicast to end users." In *Proceedings of IEEE Infocom*, June 2002.

[56] B. Y. Zhao, J. Kubiatowicz, and A. Joseph. "Tapestry: An Infrastructure for Fault-tolerant Wide-area Location and Routing." Technical report, UCB/CSD-01-1141, University of California, Berkeley, CA, USA, April 2001.

[57] S. Q. Zhuang, B. Y. Zhao, A. D. Joseph, R. Katz, and J. Kubiatowicz. "Bayeux: An architecture for scalable and fault-tolerant widearea data dissemination." In *Eleventh International Workshop on Network and Operating Systems Support for Digital Audio and Video (NOSSDAV 2001)*, 2001.

[58] C. Diot et al., "Deployment Issues for the IP Multicast Service and Architecture," *IEEE Network*, Jan. 2000, pp. 78C88.

[59] V. Roca and A. El-sayed, "A host-based Multicast (hbm) Solution for Group Communications," in *1st IEEE Int'l. Conf. Networking*, Colmar, France, July 2001.

[60] H. Deshpande, M. Bawa, and H. Garcia-Molina, "Streaming Live Media Over Peers," Stanford Univ., Database Group, Submitted for publication, 2002.

[61] D. A. Tran, K. A. Hua, and T. Do, "Zigzag: An efficient peer-to-peer scheme for media streaming," in *Proceedings of the IEEE INFOCOM*, 2003.

[62] L. Mathy, R. Canonico, and D. Hutchison, "An Overlay Tree Building Control Protocol," in *3rd Intl. Wksp. Networked Group Commun.*, London, U.K., Nov. 2001.

[63] L. Mathy et al., "Scalable Adaptive Hierarchical Clustering," *IEEE Commun. Lett.*, vol. 6, Mar. 2002, pp. 117-119.

[64] A. EL-SAYED, V. ROCA and L. Mathy, "A Survey of Proposals for an Alternative Group Communication Service," *IEEE Network Magazine special Issue on Multicasting: An Enabling Technology*, January/February 2003.

[65] J. Liebeherr, M. Nahas, and W. Si, "Application-Layer Multicast with Delaunay Triangulations," *IEEE GLOBECOM*'01, Nov. 2001.

[66] S. Zhuang et al., "Bayeux: An Architecture for Scalable and Fault-Tolerant Wide-Area Data Dissemination," *11th Int'l. Wksp. Net. and Op. Sys. Support for Digital Audio and Video*, June 2001.

[67] D. Waitzman, C. Partridge, and S. Deering. "Distance Vector Multicast Routing Protocol." RFC 1075, November 1998.

[68] S. Banerjee and B. Bhattacharjee. "A comparative study of application layer multicast protocols." In Submitted for review, 2002.

[69] S. Androutsellis-Theotokis, "A Survey of Peer-to-Peer File Sharing Technologies," White Paper, Athens University of Economics and Business

[70] I. Clarke, O. Sandberg, and B. Wiley. "Freenet: A distributed anonymous information storage and retrieval system." In *Proceedings of the Workshop on Design Issues in Anonymity and Unobservability*, Berkeley, California, June 2000.

[71] I. Clarke, TW. Hong, O. Sanberg, and B. Wiley. "Protecting free expression online with freenet." *IEEE Internet Computing*, 6(1):40–49, January-February 2002.

[72] Q. Lv, S. Ratnasamy, and S. Shenker. "Can heterogeneity make gnutella scalable?" In *Proceedings of the 1st International Workshop on Peer-to-Peer Systems (IPTPS '02)*, MIT Faculty Club, Cambridge, MA, USA, March 2002.