Load Balancing in Multihomed Stub Networks

A Thesis Submitted

for the award of the degree of

Doctor of Philosophy

by

Ashok Singh Sairam

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

INDIAN INSTITUTE OF TECHNOLOGY GUWAHATI

GUWAHATI - 781039, ASSAM, INDIA

May 2009

STATEMENT

I do hereby declare that the matter embodied in this thesis is the result of investigations carried out by me in the department of Computer Science & Engineering, Indian Institute of Technology Guwahati, India, under the supervision of Prof. Gautam Barua.

In keeping with the general practice of reporting scientific observations, due acknowledgments have been made wherever the work described is based on the finding of other investigators.

May, 2009 Ashok Singh Sairam

CERTIFICATE

*This is to certify that the work described in this thesis entitled "**Load Balancing in Multihomed Stub Networks**" submitted by **Ashok Singh Sairam** a research student in the department of Computer Science and Engineering, Indian Institute of Technology Guwahati, for the award of the degree of* **Doctor of Philosophy**, *is a record of an original research work carried out by him under my supervision and guidance. The thesis has fulfilled all requirements as per the regulations of the institute. The results embodied in this thesis have not been submitted to any other university or institute for the award of any degree or diploma.*

Gautam Barua
Professor,
Department of Computer Science & Engineering,
Indian Institute of Technology Guwahati.

May, 2009
Guwahati
Assam, India.

Acknowledgments

Acknowledging, everyone who have influenced me during the realisation of this thesis is not possible in such a limited space. The few names that I mention here are the ones I could remember at this time of writing. First of all I would like to express my gratitude to my adviser, Prof. Gautam Barua who inspired me with the idea of this project and who has been a constant source of motivation and guidance. My adviser is a person who literally has his breakfast, lunch and dinner in different cities on the same day. Despite his busy schedule, he has always given me the time whenever I needed his input and guidance. I am indebted to the members of my doctoral committee for their valuable inputs and suggestions.

I feel myself extremely lucky to be associated with the Computer Centre of IIT Guwahati, which had allowed me access to systems and data which were crucial to my project. I would also like to thank the faculty members of the department, in particular Prof. Sukumar Nandi for his constructive criticisms and pestering. I appreciate all my colleagues and to the numerous people with whom I have interacted during the course of the project.

Lastly, I am most indebted to my family for bearing with me during these long years of the project.

Abstract

While there has been a steep growth in Internet bandwidth, international bandwidth prices have declined sharply. This has led to a trend where large enterprises and educational institutes continually increase their Internet bandwidth and subscribe to more than one ISP (multihoming) to increase resilience. Higher speeds will of course mean better performance but beyond a point, the improvement in performance does not match the increase in speed. There are a number of factors that affects download speed. One such factor is that Internet Service Providers (ISPs) over-provision their resources and there can be periods when all links of an ISP are saturated. However, if the enterprise is multihomed then it can route its traffic through a less congested path. In a multihomed environment traffic along one path can experience congestion even when other paths are under-utilised. In this dissertation we propose a solution where an end network monitors the bandwidth of egress links and re-distributes the traffic along less congested paths. Our main goal is to improve the download speed by load balancing the incoming traffic. Controlling incoming traffic is difficult since an end network has little control over it. The time-scales considered in this dissertation are small (5 minutes). Internet traffic is known to exhibit highly chaotic behavior at such time-scales. Our attempt to load balance traffic for such small time-scales makes the problem even more challenging.

The solutions proposed are based on controlling the egress route of a node. The key advantage of our scheme is that we try to improve performance by optimising the resources

that are at the disposal of the local network and do not rely on any assistance from uplink networks. Moreover, we ensure that when traffic is switched from one link to another, ongoing connections continue uninterrupted. This makes our route control schemes transparent to end users. In order to make our approach practical, easily deployable and acceptable to a larger audience, we have also consciously tried to use existing tools and standards. In the first part of our thesis, we propose a simple route control mechanism to load balance Internet traffic in a multihomed local area network (LAN). We deploy a prototype of our scheme in an actual campus network. This experiment emphasizes the following two points. First, it demonstrates how our scheme can be actually implemented in a real network. Secondly, we show that load balancing the traffic improves the overall Internet round-trip times (RTTs), throughput and reliability. Our experiments show that multihoming can improve the round trip times by as much as 15 percent. We also demonstrate that performance improvements can be achieved even with a very coarse granularity of load balancing. We showed how an overall performance improvement can be realised by balancing the traffic of Internet proxy servers.

Networks cannot accurately estimate their link metrics as it has limited knowledge about user demands, available network resources and routing policies of other competing networks. A major challenge in interdomain traffic engineering is due to the level of uncertainty faced by networks. As the size of the network scales these limitations become more prominent. Moreover in larger networks like that of a stub Autonomous System (AS), the intradomain path from a node to an egress link can be several hops and it can be a source of bottleneck too. In the later part of our work we propose route control techniques to load balance traffic for such large end networks. We solve the problem in two steps. In the first phase, we make unrealistic assumptions that the input traffic is known and bandwidth measurements are accurate (offline model). Even with such *strong* assumptions we show that the problem is NP-complete. In the second phase we assume a realistic net-

work environment where input traffic is unknown and network measurements are inaccurate (online model). Leveraging upon our solutions proposed for the offline model, we design a number of heuristics. During each period users are provisionally assigned egress routes. Based on the input traffic seen in the succeeding period, the egress routes are re-assigned (recourse action). This process continues recursively. The performance of our techniques was evaluated in a static intradomain traffic environment as well as in a dynamic network environment. The simulations were performed using both synthetic as well as actual traffic traces. We found that when no load balancing mechanisms were used, deviation of the utilisation of the links from their *ideal* value ranged from 15 - 30 percent. Employing our route control schemes reduced the deviations by about 40 percent.

Ideally, the problem of re-assigning traffic should be dealt at the global level where a manager can see the traffic state of the whole network. Hence the route control schemes proposed have a centralised approach. However, centralised routing techniques cannot scale and due to the geographical distance between nodes collecting all the required information at a central location will prove costly. Moreover, a distributed architecture is not consistent with the distributed design philosophy of the Internet. Therefore, in the final part of our work we propose distributed versions of our route control schemes. Using probability analysis we show that the *expected* performance of our distributed scheme is comparable to that of the centralised one. Empirical results are used to further validate our theoretical analysis. The distributed route control schemes were validated by repeating the experiments performed for the centralised algorithm. After each simulation we draw a parallel between the distributed algorithm and the greedy approach. In all the experiments that were performed, we found that the performance of the distributed schemes is comparable to the centralised techniques both in terms of traffic load balancing, traffic re-assignments and improvement in RTTs.

For

Vijoylakshmi

And

Aradhana

Contents

List of Figures

List of Tables

LIST OF COMMONLY USED ACRONYMS

DSL	Digital Subscriber Line
ADSL	Asymmetric Digital Subscriber Line
AS	Autonomous System
ISP	Internet Service Provider
IGP	Interior Gateway Protocol
OSPF	Open Shortest Path First
OSPF-TE	Traffic Engineering Extensions to OSPF
BGP	Border Gateway Protocol
ICMP	Internet Control Message Protocol
QoS	Quality of Service
NAT	Network Address Translation
TTL	Time-to-Live
RTT	Round-trip Time
SNMP	Simple Network Management Protocol
LAN	Local Area Network
WAN	Wide Area Network
RUS	Random User Selection

Chapter 1

Introduction

In the early 1990s it was predicted that Internet traffic will double every three months. Although such predictions were later proved to be false, Internet traffic continues to grow and change over relatively short time ([87]). Internet traffic has been growing at nearly twice the rate since the year 1997 ([86]). Along with the growth of Internet traffic, end user speeds have also improved substantially over the same period. Internet speeds have increased with the migration of users to cable modem and DSL connections, backbone links are upgraded and service providers install new faster routers to handle high volume of traffic. With the increasing reliance on Internet for a wide range of task, the trend is moving toward providing even faster connectivity. Today Internet service providers are rapidly moving toward providing Ethernet or fibre everywhere.

When access speeds are limited, Internet performance will directly depend on the connection speed. For example when a dial-up user moves over to a DSL link, the difference in performance will be significant. But will the performance improve as we keep on increasing the connection speed? Higher speed will of course mean better performance but the improvement in performance will not be commensurate to the increase in speed beyond a certain point. This is because the bottleneck may not be in the last mile but beyond the

provider's network. However to exactly state at what point the disparity between performance and speed will occur will be difficult to state due to the dynamics of the Internet traffic. A study in this direction can be found in [22]. In such a situation a possible first step toward route control would be subscribe to the Internet from more than one provider (multihoming). Multihoming provides an elegant disaster recovery scheme, while improving performance. The second step is to add *intelligent route control* or *smart routing* ([65], [42]). Intelligent route control aims to pick up the best route on the fly for users with two or more ISPs. Studies have shown that the performance benefits of multihoming can be as high as 25 percent ([21]).

In a pioneer work on distributed computer systems, a design principle called *end-to-end* was proposed ([101]). The idea was to keep the core of the computer network simple and allow *intelligence* to reside at the edge of the network, closer to the end user. This principle has proved to work well in many applications and distributed systems. For instance the *end-to-end* principle is one of the central design principles in TCP. *Intelligent route control* adheres to the end-to-end principle, since these tools are deployed at the edge of the Internet to address the fundamental issue of Internet performance and cost barriers. The focus in this dissertation is to develop route control techniques for a multihomed network to load balance the incoming traffic across its multiple links without disturbing the core of the Internet.

1.1 Factors affecting Internet Performance

A network that only produces or consumes packets but do not transit packets of other network is called an end network or stub. Internet performance in an end network is affected by congestion and latency between the user and remote server, performance of traffic engineering techniques employed and performance of the remote server and its back-end databases. An end network may have a dedicated link between itself and its ISP, but

beyond a certain point its Internet traffic flows will share network resources with millions of other flows. Usually last mile connectivity links have much lower speed than those of backbone links. The issues that affect download speed can be grouped into two - *local* and *remote*. Download speed limited due to last mile connectivity of the source, ill-configured software stack and errors in router configuration are *local factors*. Such issues can be easily overcome by locally updating the system. Remote factors on the other hand are those over which an end network has no control. Data transfer rates limited due to congestion between the source's and destination's provider, congestion in the last mile connectivity of the destination, performance of the remote server and its back-end databases are *remote factors*. Overcoming performance bottlenecks due to remote factors will require network-wide upgradation and management. With many competing ISPs such global coordination can be very difficult. Thus, with limited knowledge about network topology, available bandwidth and routing policies of other networks, a stub network cannot guarantee end-to-end performance improvement. Under such circumstances, the best a stub network can do is to try to fully utilise the resources at its disposal.

1.2 Internet Traffic Characteristics

Characterising Internet traffic is difficult due to its heterogeneous and time-varying nature. Nevertheless, researchers have identified certain aggregate characteristics about Internet traffic. In this section we summarise some of the important characteristics that are relevant to our work.

1.2.1 Bi-directional but Asymmetric

Previous as well as more recent studies of Internet traffic ([110], [108]) reveal that TCP packets still dominate Internet traffic. Although many of the applications used on the Internet generate bi-directional flow of data, volume of data is heavier in one direction than the other. Proliferation of peer-to-peer applications could have affected this asymmetry, but

these applications could not take off in a large scale due to legal and network considerations ([86]). The asymmetric nature of Internet traffic is likely to continue for the foreseeable future. The growing popularity of ADSL connections is a proof of this. Organisation, therefore, try to optimise either the traffic that enter or leave the network based on its business interest. Content providers that host a lot of web or streaming servers will have several customers wanting to download traffic from its network. Such networks will try to optimise the traffic that leaves their network. On the other hand access providers that serve small and medium enterprises will have users that primarily want to download traffic from the Internet. In access networks the main objective is to optimise the way traffic enter the networks.

1.2.2 Long-Range Dependent but Chaotic for Small Time Scales

Leland et. al. in a pioneering work ([81]) established that Ethernet traffic patterns are self-similar (SS) and long-range dependent (LRD). The discovery of these properties was hailed as a major breakthrough among the research community and their applicability have been extensively studied. Later it was found that these SS and LRD properties are ubiquitous: many network traffic including Internet traffic exhibit these properties ([41]). Thompson et. al. ([108]), in their study on wide area traffic, shows that Internet traffic levels follow a 24-hour pattern. The issue of self-similarity has been addressed from many aspects including its impact on network performance, modeling techniques and its causes. From a statistical viewpoint, self-similarity implies studying the variations in the signal strength linearly with the considered time scale. Interdomain traffic variability has shown to become limited over timescales larger than one hour or so ([99]). This suggests that from a control perspective, interdomain traffic engineering should focus on timescales larger than one hour or so.

However, Internet traffic is known to exhibit different scaling characteristics at different

time-scales. Internet traffic behaviour when analysed on small time-scales (typical round-trip time) transforms rather abruptly to a more complex behaviour ([60], [49]). At such small time scales Internet traffic seems to have multi-fractal scaling behaviour ([48]). Jiang et. al. ([60]) showed that such burstiness of Internet traffic can extend to the round trip time of dominant TCP flows. Due to this complex nature of Internet traffic no traffic model has been able to completely capture the behaviour of Internet traffic. One of the issues that we try to address in this dissertation is how to make bandwidth provisions in the absence of a formal traffic model for small time scales. In this dissertation our goal is to propose route control techniques without making any presumptions about the network traffic.

1.2.3 Elephants and Mice

Studies on Internet traffic usually track user (host) activity by observing the user as an IP address ([38]). In this work too an IP address is treated as a user. Internet traffic is always a mix of flows from a variety of applications. A flow is usually classified by its size or duration. Studies show that user traffic flows have usually low throughput and are short-lived. This low throughput, short-lived connections are referred to as mice in literature. Due to the presence of mice, the number of sources that become active or passive is high for small time scales ([99]). While it is true that majority of the Internet flows have low throughput, a small fraction of users account for major volume of the traffic (referred to as elephants). The common observation is that 10 percent of users (elephants) account for 90 percent of the traffic ([99],[33], [46]). The presence of this property implies that by moving a few user among the egress links of an enterprise we can achieve a fair degree of traffic load balancing.

1.3 Multihoming

A network or a set of networks with a unified administrative routing policy is called an autonomous routing domain (ARD). Examples may be an internal ISP network, corporate

network, campus network et. al. Autonomous system (AS) also called domain is an ARD that has a unique autonomous system number ([67]). Although the notions of an AS and ARD are different, in this work the terms AS, ARD and domain are used as synonyms. The Internet has seen a substantial growth in the number of ASes. This growth has been mainly due to the increase in a large number of stub ASes. In stub ASes, the basic form of Internet access is *singlehoming*, where the end network uses a single upstream ISP to reach all destinations. Transit ASes interconnect several networks and hence they are usually connected to more than one provider (multihomed) or peer. Interestingly this trend of multihoming has been observed in stub ASes too ([104],[109]). This trend toward multihoming is only likely to increase with more interdomain traffic engineering tools and techniques exploiting multihoming to extract performance benefits ([65],[42]).

In general a multihomed network will find that traffic on one path is congested, even when traffic on other links is under-utilised. In figure 1.1, we plot the link utilisations of a two ISP multihomed stub network. As can be seen from the plot, for most of the time periods there is a substantial difference in the link utilisations although their link capacities are the same. Localised congestions can also occur anywhere in the Internet due to distributed denial of service attacks (DDoS) and Internet worms. An efficient way of re-routing traffic around such localised congestion points can be an ideal tool for networks to improve their performance. Smart or intelligent routing also referred to as route optimisation is a source-based route control technique, wherein a multihomed network distributes its traffic among multiple Internet links ([65],[42]). Such source-based routing is also called *selfish routing* since a user tries to optimise its metric without considering its impact on other traffic. In [56] it has been shown that smart-routing users can coexist well with very little interference to other users. The solutions proposed in this dissertation can be categorised in the broad realm of smart routing, although our objectives and optimising criteria are very different.

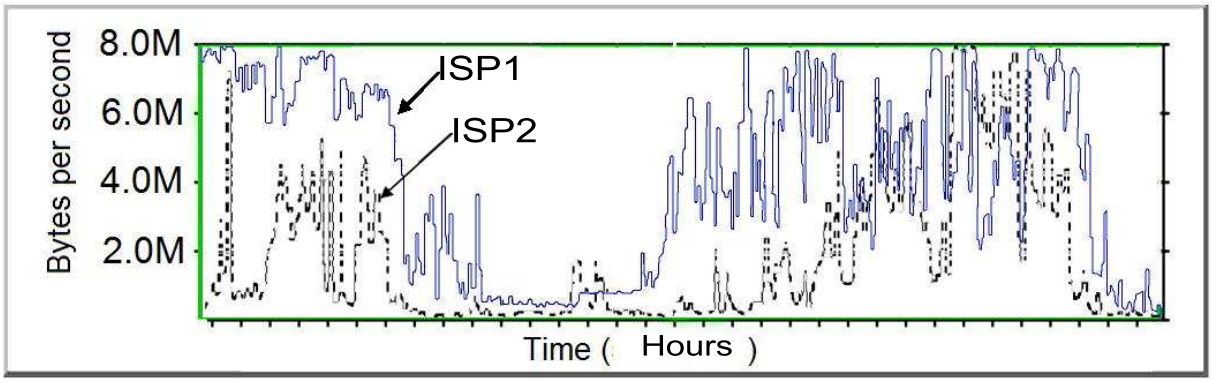


Fig. 1.1 Comparison of link utilisations of a 2-ISP stub network.

1.4 The Problem and our Approach

Figure 1.2 shows the incoming and outgoing traffic on the egress link of a campus network. As can be see from the graph the volume of incoming traffic is relatively very high. Studies have shown that in access networks, the incoming traffic is more than five times of the outgoing traffic ([99], [11]). Most of the work on traffic engineering attempt to control the flow of outgoing traffic ([109], [64], [78], [85]). Some of these outbound traffic engineering techniques will no doubt influence the incoming traffic, but this aspect has not been studied. Inbound traffic control is difficult because it involves influencing the behaviour of the remote destination. Secondly, an end network has very limited or no knowledge about the topology, available bandwidth and user demands of other networks through which its traffic transits. This uncertainty makes load balancing of interdomain traffic very challenging. Thirdly, network traffic is routed based on destination. With destination based routing, optimising inbound traffic becomes extremely difficult, since network announcements cannot be done individually for all the different destinations.

The fundamental question that we seek to answer in this work is:

How can a multihomed, stub access network optimally utilise the bandwidth of all its egress links to improve its Internet experience?

In this dissertation we first review the existing traffic engineering techniques including BGP

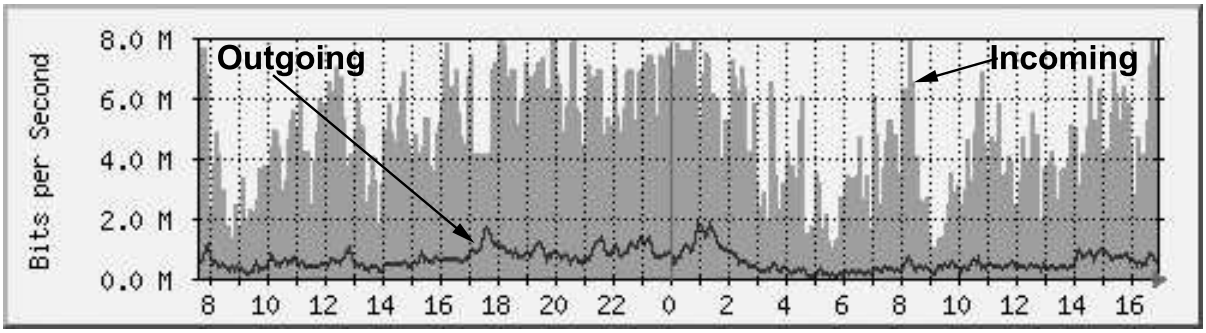


Fig. 1.2 Internet traffic of a campus network.

([112]), the current de-facto interdomain routing protocol. We find that these traffic engineering tools concentrate on improving the end-to-end performance rather than distributing the traffic. Moreover, these tools having been primarily designed for outbound traffic, they are not competent to handle incoming traffic. Based on our quantitative evaluation of the different traffic engineering techniques we conclude that incoming traffic cannot be controlled by directly acting on the traffic. We, therefore, propose to control the flow of the incoming traffic by acting on its corresponding outgoing traffic. In the solution that we propose, an end network monitors the incoming bandwidth on all its egress links. In order to ensure that utilisation of the incoming bandwidth is not limited by *remote factors*, the monitoring is not end-to-end but only on the path between the end network and it's ISP. These paths are called *connecting paths* or *connecting link*. The congestion (or restriction in available bandwidth) is assumed to be only on these paths. Tools that can be used to monitor such paths is identified in chapter 3. When the end network observes that a particular egress link is congested, it does not try to control the routing of its ISP. Rather it re-routes some of its incoming traffic to a relatively less congested egress link. As the focus of our work is on access networks, requests for downloads will primarily originate from within the network. Internet traffic being primarily in the form of request-response pairs, the re-routing is achieved by scheduling the outgoing request appropriately. The only requirement for our scheme to work is that the request and response of a traffic flow must follow the same egress route. The network models which can be used in our solution are

given in chapter 2. Our proposed solutions encompass all types and sizes of end networks. The proposed route control techniques neither require any modification to existing routing protocols nor support for Internet-wide infrastructure. This will make our techniques acceptable to a wider range of audience.

1.4.1 Dissertation Outline

The rest of this dissertation is organised as follows. In chapter 2, we review traffic engineering techniques used in different scenarios - intradomain, interdomain and Internet-wide. Putting together the information gained from our review of the different traffic engineering techniques, we summarise a general model for controlling incoming traffic. We then present network models which can be used in our solution. The first step toward load balancing traffic is to monitor and detect congestion on the links. In chapter 3, we survey the various tools available for estimating bandwidth and devise a methodology for measuring available bandwidth of the *connecting path*.

We attempt to solve the problem of managing incoming traffic at two levels. In chapter 4, we tackle the problem of managing incoming traffic in small end networks (LANs). We implement a prototype of our approach in an actual campus network and demonstrate how a multihomed LAN can improve its Internet experience. In the second part of our work (chapter 5), we consider larger end networks like a stub AS. We highlight the complexity of managing incoming traffic and go on to prove that the problem is NP-complete. We then propose heuristics and validate our proposals using real traffic traces. To keep our work in tune with standard network routing protocols, in chapter 6 we propose distributed versions of our heuristics. We prove that the *expected* performance of the distributed heuristics is comparable to that of the centralised approach proposed in the earlier chapters. Empirical results are used to further validate our claim. Finally, in chapter 7 we present our conclusions and future direction of our work.

Chapter 2

Background

Due to large scale deployment of fibre cables, there has been a huge growth in Internet bandwidth and at the same time prices have gone down. However, as we can see from table 2.1 ([89]), the growth in Internet bandwidth has not been able to keep up pace with demand. There are several factors that fuel the growth of Internet traffic. While Internet penetration rates have saturated in developed countries ([3]), these rates continue to grow rapidly in other countries. The second factor that fuels Internet growth is due to increasing intensity of Internet use among existing users. IP routing policy of choosing the shortest path does not always select the best path or result in good network utilisation. Optimising and improving Internet performance has and will remain a hot topic of research for the foreseeable future.

Traffic Engineering is the process of controlling how traffic flows through one's network so as to optimise resource utilisation and network performance ([111]). A detailed discussion on issues surrounding traffic engineering in IP networks is given in [43]. In this chapter we look at some of the prevalent traffic engineering techniques from a perspective of our research work. Based on the scope of application, we study traffic engineering techniques under three groups: (i) Intradomain traffic engineering (ii) Interdomain traffic engineering

and (iii) Internet-wide traffic engineering. Based on our quantitative evaluation of the different traffic engineering techniques, we propose a general model for managing incoming traffic. Network models that will support load balancing of incoming traffic are discussed. Finally we review those works that are related to ours in terms of the traffic input, network model and goal.

Table 2.1 Annual Growth in Internet Traffic and Bandwidth.

	2003 - 04	**2004 - 05**	**2005 - 06**
Average Traffic	104%	50%	75%
Internet Bandwidth	45%	43%	47%

2.1 Intradomain Traffic Engineering

Inside a domain, a link state protocol (also called Interior Gateway Protocol, IGP) like OSPF ([72]) is used to flood the topology information to all routers within the routing domain. Best end-to-end paths that optimise certain metric chosen by the network administrator are computed locally at each of the routers. Traffic is then routed through these pre-computed paths. Traffic engineering goal within a domain is to improve traffic distribution throughout the entire network. The IGP *link cost* metric greatly influences utilisation of intradomain links. A number of intradomain traffic engineering works has been centered on determining the optimal link cost weight. Techniques to set optimal OSPF weights for a known traffic matrix is proposed in [28], [52]. A proposal to distribute traffic over equal cost multi paths is given in [107]. Typically, link cost is set to inverse of the link bandwidth. Default OSPF link cost setting on Cisco routers is described in [6].

Interdomain traffic engineering on the other hand refers to managing the route of packets that flow across domains. Although intradomain and interdomain traffic engineering are usually treated in isolation, there is a strong correlation between the two ([96], [73]). Interdomain traffic engineering can greatly impact traffic on intradomain routes ([19]). At

the same time internal network traffic also have a substantial contribution on the overall network latency ([90]). A large end network like an AS is usually an interconnection of smaller networks spread over a large geographical area. The influence of intradomain traffic on interdomain traffic cannot be ignored, especially for such large end networks. Thus one of the objectives that we consider in our problem while selecting an egress route is the intradomain link cost. The periodic link state updates of intradomain protocols do not include any traffic engineering metrics. To facilitate intradomain traffic engineering, the Internet Engineering Task Force (IETF) has proposed extensions to link state protocols. Such traffic engineering extensions primarily mean piggy-backing some performance metrics along with the periodic link state updates. For example, OSPF-TE ([44], [24]) adds traffic engineering capabilities to OSPF by incorporating available bandwidth information, hop count etc. along with the link state updates. In this work, the intradomain protocol deployed is assumed to be OSPF-TE. Intradomain link costs are set proportional to available bandwidth of the links.

2.2 Interdomain Traffic Engineering

Most of the work after the concept of traffic engineering was introduced concentrated on intradomain traffic engineering. However, with the rapid growth of Internet traffic, researchers realised that interdomain resources are also a frequent source of bottleneck. Interdomain traffic engineering requirements are driven by the motivation to balance traffic on the multiple egress links of a network, reduce cost of carrying traffic and optimise use of resources. BGP ([112]) is the current de-facto interdomain routing protocol. Understandably, majority of the work on interdomain traffic engineering revolves around BGP. Yet, there are also a number of interdomain traffic engineering techniques that do not involve BGP. In this section, we initially look at techniques that function independent of BGP. Then we study in detail interdomain traffic engineering with BGP.

2.2.1 Edge-based Bandwidth Managers

The Internet link of an enterprise is usually subscribed by a number of users. Some of these users may over-subscribe and thus have an influence on the overall performance. In order to fairly distribute the resource among the users, a bandwidth manager is installed at the edge of the network to control the incoming and outgoing traffic. The most popularly used edge-based managers are policy-based. A network administrator defines policy rules on the basis of which network packets are matched. Traffic shapers ([97]) identify and categorise specific network traffic and then constrain each category to use no more than a specified amount of bandwidth, queuing the excess packets for later transmission. Traffic policers are similar to traffic shapers but the excess traffic is dropped or marked. Such policy-based managers will not be effective on inbound traffic since they act on the traffic after it has already traversed the egress link and consumed the bandwidth.

Another class of bandwidth managers called dynamic bandwidth managers track the network traffic on a long-term basis and then depending on the average usage, allot bandwidth to users. This scheme, however, cannot meet the short-term burst requirement of users. Surveys of traffic engineering techniques are available in the literature ([64], [78]). In multihomed networks, the general principle used is to find the best outbound route to reach a destination based on active or passive measurements. A number of tools both from the research community ([21], [57]) as well as commercial ones ([4], [9], [1], [10] and [8]) are available. In general these tools aim to improve end-to-end performance rather than distributing the Internet traffic among egress links. For inbound traffic, DNS cycling technology ([21]) is used, but such techniques for inbound traffic control are coarse and does not direct inbound traffic to the least loaded link.

2.2.2 Rate Control Techniques

Internet traffic being mainly TCP, another way we can control incoming traffic is to try to influence the behaviour of the remote TCP sender. These techniques are collectively referred to as rate control techniques. The sending rate of a TCP source is determined by its window size, round trip time and rate of acknowledgments. A TCP receiver tries to influence the sending rate of the source by modifying its ack number and receiver window fields in acknowledgments and by modulating the acknowledgment rate. Three methods that are widely used are - Window Sizing, ACK Pacing and MSS Shrinking. Huan-Yun Wei et. al. ([59]) have done a detailed analysis of these methods. Karandikar et. al. ([74]) proposed a hybrid approach that combines Window Sizing and ACK Pacing. Another way a TCP receiver can influence the sender is to advertise a zero window size and force the sender into the zero window probe (ZWP) mode ([55]). During this mode the sender will freeze all re-transmit timers, enter a persist mode and send probes until the receiver window opens up. ICMP, an error reporting mechanism included with all TCP/IP implementations is also used to induce the behaviour of the TCP sender. ICMP *Source Quench Message* can be used to inform the sender to reduce its transmission rate ([79]). However, the use of ICMP is crude and not effective.

Rate control techniques are used when the demand for network resources far exceeds its capacity. Though congestion control mechanisms are necessary and powerful, they are not sufficient to provide good performance under all circumstances. These techniques try to control congestion by limiting the amount of data entering the network whereas our goal is to alleviate congestion by spreading data more evenly. Rate control techniques should be tried as a last resort to control congestion. The study of such active control mechanisms is beyond the scope of the thesis.

2.2.3 Egress Route Selection

Selecting different egress routes for Internet traffic flows can have diverse effects on the resource utilisation of an end network. Egress route selection techniques can react swiftly to traffic dynamics. The principle is to select egress routes based on an optimisation routine and network-wide performance objective. A straight-forward application of egress router selection would be to manually configure each router in a domain with a fixed ranking of egress points for each destination. Such fixed ranking of egress points, although robust, cannot adapt to traffic dynamics and topology changes. In order to account for traffic dynamics, we need to dynamically rank the egress routers. A number of works on dynamic ranking of egress routes is available in the literature ([93] and [77]). A more detailed survey on egress route selection techniques is given in chapter 5. In this dissertation we assume that the source network has no control on network routing, except on the choice of exit points from a set of possible paths. In this context, our work is similar to the egress route selection problem.

2.2.4 Traffic Engineering with BGP

Although BGP was not designed with traffic engineering goals in mind, nevertheless a large number of works on interdomain traffic engineering involves tweaking BGP. The operation of BGP is fairly simple. There is no broadcasting of information. A BGP router establishes a TCP session with another BGP speaking router. Routes are first advertised when the session is established. Thereafter, routes are updated when they change. Routes for destinations that become unreachable are removed. A route advertisement indicates the reachability of a network. It consists of a network address and a mask representing a block of contiguous addresses. Figure 2.1 shows a simplified BGP network of five ASes. In this topology, the stub AS, AS1, will announce reachability of its own address 16.1.2.0/24. There are two variants of BGP. The eBGP variant is used to announce reachable prefixes between neighbouring ASes. The iBGP variant is used to announce reachability within a

domain.

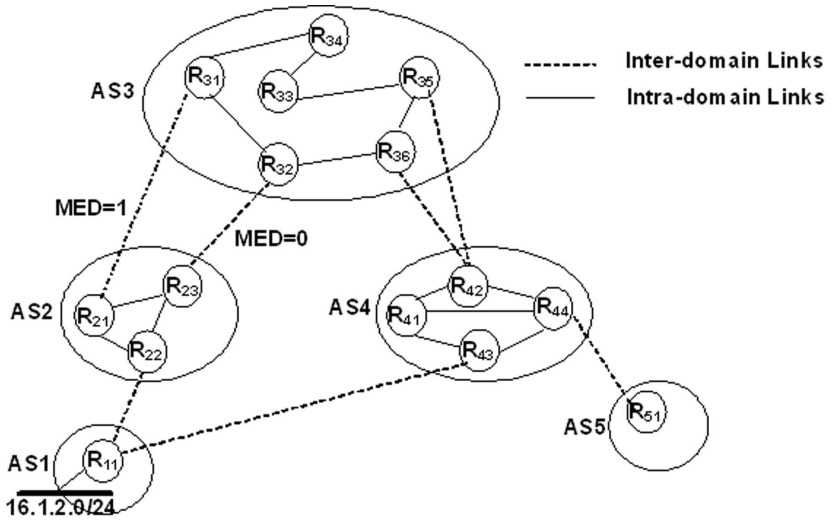


Fig. 2.1 A simplified Internet topology.

Working of a BGP Router

A BGP router with several BGP peers will accept advertisements from each of the peers. To begin with, the BGP router applies an input filter for each BGP peer and selects only acceptable advertisements. For example, a BGP router may accept advertisements containing trusted ASes only. In case there is more than one BGP peer, the first step will usually result in multiple routes for a given destination prefix. The second step in the BGP decision process is to select from among the multiple routes for a given destination prefix, a single best route. The routing table will be populated with the next-hop of these best routes. Thirdly from among these best routes the router will select routes that will be advertised to other BGP peers. At most one route will be advertised for each reachable destination. Routers learned from upstream providers or peers are usually not exported because there is no economic incentive to transit traffic from providers and peers. However, routes learned from customers will be exported to all other BGP speakers. In the second step, a BGP router needs to select one route from among several routes to reach a destination. To allow operators a more systematic approach in this decision process, in the BGP announce-

ments along with destination prefix, several additional attributes are advertised. In table 2.2 we list the important attributes in order of their preference. A common technique of traffic engineering with BGP involves manipulating the value of these attributes ([47], [29]).

The first attribute *local-pref*, is an administrative cost specifying the preference among the different routes toward a given destination. The route with the highest value of local-pref is considered the best one. This attribute is set upon receipt by the local router. The second attribute *AS path* is the list of AS through which the announcement passed. The second rule, therefore, is to select the route with the smallest AS path. The third attribute *origin type* concerns how the originating AS learned about the route. The rule here is to prefer IGP over EGP (Exterior Gateway Protocol, this protocol is nowadays no longer used). The fourth attribute *multi exit discriminator* (MED) can only be used between a pair of AS that share more than one route. This attribute is used to prefer one egress point over the other. The rule is to prefer the route with lowest MED value. Routes that do not have a MED attribute are assumed to have the lowest MED value. The fifth rule prefers route learned through eBGP over those learned over iBGP. The sixth rule prefers the next hop to a destination that has the lowest IGP path cost. The objective of this rule commonly referred to as *hot-potato routing*, is to get rid of a packet as soon as possible so that minimal resource within the domain is consumed. The last rule is to prefer the route with the lowest *router id* (IP address of the BGP peer). This rule is also called the tie breaking rule, because in case the first six rule has failed to obtain a single best route, it will do so.

Controlling Outbound Traffic

The first technique to control outbound traffic is to use local-pref attribute. This optional attribute is used inside a domain only. A BGP speaker upon receiving an advertisement inserts the local-pref attribute before distributing the route via iBGP to other routers of

Table 2.2 Attributes of a BGP advertisement.

Step	Attribute	Which AS controls?	Effect of TE
1	Highest local-pref	local	outbound
2	Shortest AS path	peer	inbound
3	Lowest origin type	neither	neither
4	Lowest MED	peer	inbound
5	eBGP over iBGP	neither	neither
6	lowest IGP cost	local	outbound
7	lowest router id	neither	neither

the domain. For example the local-pref attribute could be set in such a way that a high bandwidth link is preferred over a low bandwidth link. Another practice to load-balance outbound traffic is to set local-pref value based on passive or active measurements. A second technique often used by large transit providers is to invoke *hot-potato routing*. Most of the BGP-based interdomain traffic engineering work is devoted to controlling the outbound traffic ([109], [83], [106]). Commercial solutions to manage Internet connectivity based on BGP-based techniques are also available ([10], [2]). In this dissertation our main goal being to control incoming traffic, we do not go into the details of these approaches.

Controlling Inbound Traffic

The key challenge in inbound traffic control is that the local domain needs to influence the behaviour of the remote domain, which in turn may have conflicting traffic engineering goals. The first technique is to use the MED attribute. This attribute can be used between a pair of domains connected via multiple peering links. For example in figure 2.1, AS2 has two external links with AS3: $R_{21} - R_{31}$ and $R_{23} - R_{32}$. If AS2 finds that the link $R_{21} - R_{31}$ is congested, then it could reduce the traffic on the link by increasing the MED value, router R_{21} advertises. This will make the other link $R_{23} - R_{32}$ more preferable to AS3. A common method is to set the MED value equal to the IGP cost toward the next-hop. The drawback is that this technique can be used only between ASes that have multiple links and it requires a bilateral agreement.

The second technique an AS can use to influence incoming traffic is called AS path prepending. The idea is to reduce the preference of a link by prepending multiple copies of its AS number to the AS-path, thereby artificially inflating the AS-path length. Lets again consider figure 2.1. The stub AS, AS1, receives traffic from two links: $R_{11} - R_{22}$ (AS2) and $R_{11} - R_{43}$ (AS4). AS1 wants to move some incoming traffic from AS4 to AS2. AS1 can effect this by prepending its own AS number on the advertisements it sends to AS4. This will make the routes of AS1 advertised via AS2 more preferable to other ASes, than to those advertised via AS4. The challenge, here is that the amount of prepending required needs to found by trial and error.

The effectiveness of the above two techniques will depend on how the destination and transit ASes have tuned their outgoing traffic. For example in figure 2.1, if AS3 has tuned its local-pref attribute such that its traffic destined for AS1 is routed through AS4, then no change in MED or AS prepending can force AS3 to route traffic for AS1 via AS2. BGP allows very limited control over incoming traffic. Therefore, proposals to control inbound traffic should be collaborative ([20]).

The third technique used is called selective announcement and it does not involve tweaking of the BGP attributes. The idea is to announce different route advertisements on different links. Consider figure 2.2, AS2 has two egress links, one with AS3 ($R_{21} - R_{31}$) and the other with AS4 ($R_{23} - R_{41}$). Further let us assume that AS2 uses two subnets within its domain: 16.1/16 and 17.1/16. In order for AS2 to balance the incoming traffic, it will announce the subnet 16.1/16 on the link with AS4 and the other subnet 17.1/16 on the other link with AS3. The problem with such an announcement is that if one of the link fails, the subnet announced through that link will become unreachable. While routing packets, an IP router always selects from the routing table the most specific route for each

packet. Taking advantage of this fact, we have another variant of selective advertisement which allows more precise control of the incoming traffic and at the same guarantees a fall back link in case of a failure. The technique is to announce a large prefix on all links for redundancy but prefer some links for part of this prefix. Continuing with our example, let us suppose that AS2 additionally announces the prefixes 16/8 and 17/8 on the links $R_{21} - R_{31}$ and $R_{23} - R_{41}$ respectively. In the case of failure of link $R_{23} - R_{41}$, hosts having prefix 16.1/16 will still be reachable through the other link $R_{21} - R_{31}$. The drawback of this approach is that it increases the size of BGP routing tables on all routers on the Internet.

Another technique used is *community-based* traffic engineering. The *communities* attribute is an optional 32 bit value used by ISPs to attach optional information along with the route advertisements. For example, pre-defined community values can be attached to set the local-pref attribute, not to include certain routes in the advertisements etc. The BGP community attribute is described in [98] and detailed survey on the utilisation of this attribute can be found in [88]. The main drawback is that this technique relies on ad hoc definition of community values and error-free manual configurations. Moreover, this technique is not scalable.

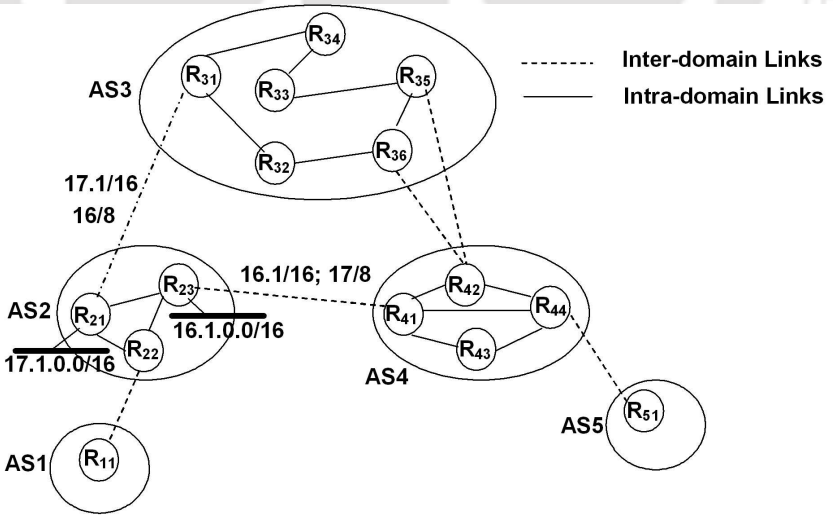


Fig. 2.2 Incoming Traffic Control in BGP.

2.3 Internet-wide traffic engineering

As we have seen in the previous sections, intradomain or interdomain traffic engineering alone cannot guarantee a good end-to-end Internet routing performance and resilience. Co-operation among all participating networks is required. To address these issues, researchers have proposed Internet-wide infrastructure called overlays. In overlay routing, virtual links are established across several networks throughout the Internet. Whenever, a network participating in the overlay finds that the Internet route to a destination as determined by BGP does not offer the expected performance, it can route the traffic via an alternate path using overlays. Research studies advocating overlay routing can be found in [102], [23]. In this dissertation our main goal is to optimise the utilisation of Internet paths that is in under the control of the local network. The study of overlay routing techniques is , therefore, beyond the scope of our work.

2.4 General Model to Manage Incoming Traffic

Putting together the information gained from our review of the different traffic engineering techniques, we summarize a general model for controlling incoming traffic (figure 2.3). The first step is to measure the incoming traffic on each of the egress links (per-link monitor). We can monitor a number of QoS parameters of the incoming traffic. In this work we monitor bandwidth, a prominent QoS parameter and try to optimise it. Based on the bandwidth utilisation of the egress links and policies specified by the network administrator, the second step is to mark the links (Link Marker). Depending on how the network administrator specifies the policy there can be different models. In this work we look at two models. In the first model (described in chapter 4), a threshold value is defined for each of the links. A link is marked if its utilisation exceeds the threshold value. In the second model (described in chapter 5) we compute an optimal utilisation value for each of the link based on total traffic entering the network and available bandwidth of the links. A link is marked

if its utilisation is above the optimal value. Finally, outbound traffic of those users whose inbound link is marked is classified on a per-user basis. The regulator then acts on the outbound traffic of some or all of these users. The action that the regulator can take are delaying or re-routing of the outgoing traffic.

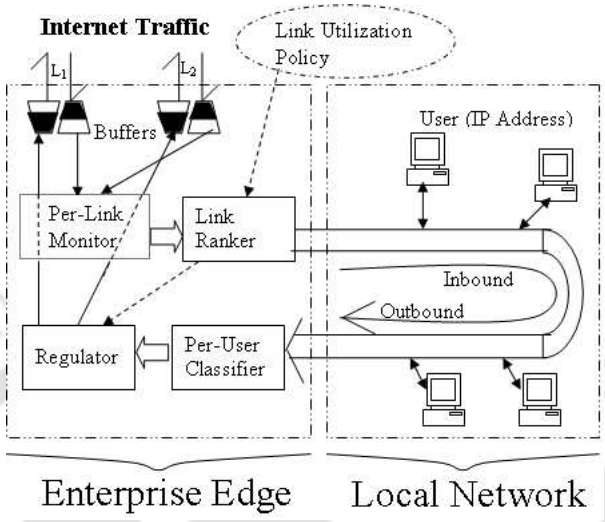


Fig. 2.3 General model to manage incoming traffic.

2.5 Network Support for Managing Incoming Traffic

In this work our aim is to load balance the incoming traffic by moving users from one egress link to the other. Incoming traffic on the egress links are monitored. If we observe that inbound bandwidth of a link is congested, a user or a group of users assigned to that link is selected and their outbound traffic is redirected to a relatively non-congested link. In this section we propose network models which can be used in our solution. The primary supports that the underlying network model should provide are:

i. Both incoming and outbound traffic of a user follow the same egress route.

ii. Allow the egress route of a user to be re-assigned dynamically.

2.5.1 NAT-based Model

The first network that we consider is one which does not have an independent IP address of its own and uses the IP address assigned by its provider. This is particularly true for *small* stub networks like LANs. The standard practice in multi-homed networks is to advertise reachability to only the IP address prefixes alloted by the ISP ([105]). Figure 2.4 show a *small* end network (LAN) connected to two ISPs. Typically users of the network will be distributed manually between the two ISPs by assigning IP address of the respective ISP. To preserve the limited number of IP addresses, users are assigned private IP address and their Internet traffic is routed using NAT or proxy servers. We illustrate our network model using two private IP address prefixes 192.168.0.0/24 and 192.168.1.0/24. To make the re-assignments easier, each user of the network is assigned two IP addresses, one from each of these pools of private prefixes. To change the egress route assignment a user merely needs to change its *default gateway*. Network Address Translation (NAT) is then used to send outbound packets with a source IP address associated by the ISP with that outbound link. The corresponding return traffic will automatically come back via the same link, because that link is the only link servicing that address range. In our example the first private IP address prefix is translated in router R1 and the second IP address prefix in router R2. NAT-based solutions are frequently used to control the flow of incoming traffic ([21], [57]). Even though a user is assigned two IP addresses, all external traffic of the user will be routed through one of the egress routes only: the default egress route of the user. In order to change the egress route of a user all we need to do is to change the default route of the user. After the egress route of a user is changed, if the user initiates a new Internet session the traffic will follow the new egress path. The question now is what will happen to the ongoing traffic flows of the user? This abrupt re-routing of the user will cause all active sessions of the user through the previous egress route to fail due to change in the source IP address. Most implementations of load balancing ignore such ongoing connections on the pretext that Internet connections are short-lived. While it is true that most of the Internet

connections are short-lived, the bulk of the bandwidth is consumed due to a few long-lived flows. Measures on how to preserve the on-going active sessions of a user are proposed in chapter 4.

An alternative implementation would be to use a single IP address prefix network-wide and then direct the default route of all users to a *centralised bandwidth manager*. The task of selecting egress routes and translating of network address is delegated to this centralised controller. The manager keeps track of which users are assigned to which routes and perform the address translations accordingly. This is the approach that we have adopted in our prototype implementation given in chapter 4.

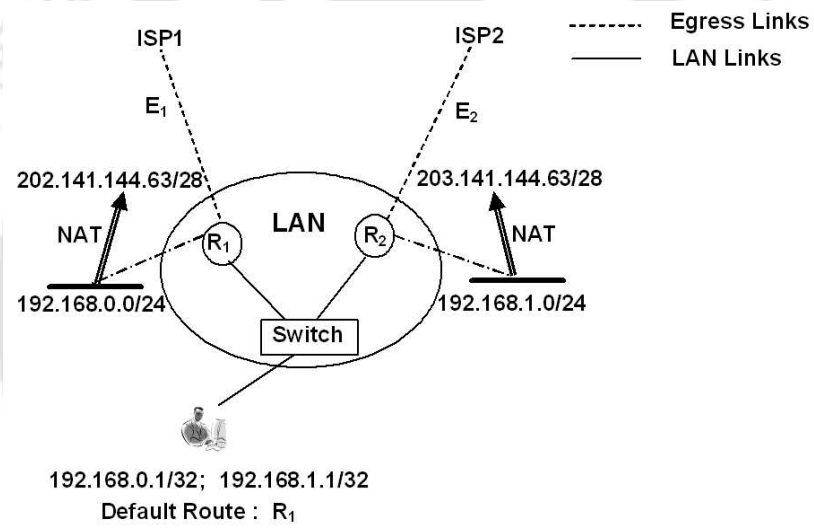


Fig. 2.4 Managing user movement in a LAN.

2.5.2 BGP-based Model

In the second model, we consider stub networks that have a block of IP addresses independent of its ISPs. While NAT-based solution by far is the most practical solution for controlling incoming traffic, NAT-based solutions are however not scalable for large domains with many hosts behind the NAT. In case of such large networks where NAT-based solutions fail to scale, we propose tweaking of BGP policies to meet our objective. BGP

being the de-facto interdomain routing protocol, changes to BGP policies are usually proposed in work that involve interdomain routing ([37]).

Let us consider a stub AS, AS1 (as shown in figure 2.5), with two egress links E_1 and E_2 connecting to two different service providers. For simplicity let us assume that the AS has two subnet addresses 17.1/16 and 16.1/16. The first network model we propose is using selective advertisement. Subnet 17.1/16 is advertised on link E_1 and 16.1/16 is advertised on link E_2. Each user is assigned two IP addresses, one from each of the subnets. As in the case of NAT-based model, all external traffic of a user will be routed through one of the egress links only (the default egress route). Once the egress route of a user is changed, traffic of new sessions initiated by the user will follow the new route. However, unlike in the NAT based model, ongoing sessions of the user will continue uninterrupted as the user will be visible through its previous egress link also. The disadvantage of this method is that fifty percent of the IP addresses are wasted. To reduce wastage in IP address we can initially allot each user a single IP address. Once a user is selected to move to a new egress route it is accordingly assigned a second IP address. The first IP address assigned to the user is revoked once there is no external traffic on that interface. A similar network model is used to control incoming traffic in [21], where a user is assigned an IP address dynamically depending on which egress route is used to reach a destination. In our route control approach, a node will require to handle connections using more than one IP address in order to preserve the active connections of a user (more details in chapter 4, section 4.4.4). Such a network model may not be able to handle this transparently without making changes to the node software.

In our second proposed network model, we use *selective sub-prefix advertisements*. We partition the original IP address prefixes into groups and then advertise these prefixes. For example, we could partition the users of the domain into groups of 64. Thus, adver-

tisements on links E_1 and E_2 will be 17.1/26, 17.1.0.64/26 $\cdots$ and 16.1/26, 16.1.0.64/26 $\cdots$ respectively. Once we change the egress route of a user group, say from link E_1 to E_2, we change the BGP route advertisements simultaneously. We announce the prefix of the user group on link E_2 and withdraw the prefix from E_1. The first advantage of this model is that there is no wastage of IP addresses. Secondly, once the route of a user group is withdrawn and re-advertised through a new route, all traffic destined toward the user (including traffic of ongoing sessions) will start following the new route immediately. This is ideally the model we prefer for an effective solution of our problem.

An issue with the use of sub-prefixes is that many ISPs filter out small IP prefixes advertisements. The reachability of the sub-prefixes announced is, therefore, not guaranteed. The challenge would be to announce sub-prefixes that are acceptable to the uplink providers. Subdividing of prefixes for load balancing, however, is a standard practice in multihomed networks ([30]). The other issue is with regard to time taken for route convergence. BGP route convergence typically range from seconds to tens of minutes ([80]). The frequent route updates required in our model may cause route convergence problems. BGP route convergence time has always been a source of concern and several techniques have been proposed to reduce it ([32]). The heuristics that we propose proceeds in periods (each period is 5 minutes). During each period it is expected that some route updates will be required. However, our algorithms are designed in such a way that once a user is moved it is not considered for re-assignment for the next several periods. This means routes of users that have been moved will converge before it is selected again for re-assignment. Thus delay due to BGP convergence will not have a serious impact on the performance of our heuristics. Moreover, tools have been developed ([47]) which allow network operators to predict flow of traffic due to changes in BGP policies.

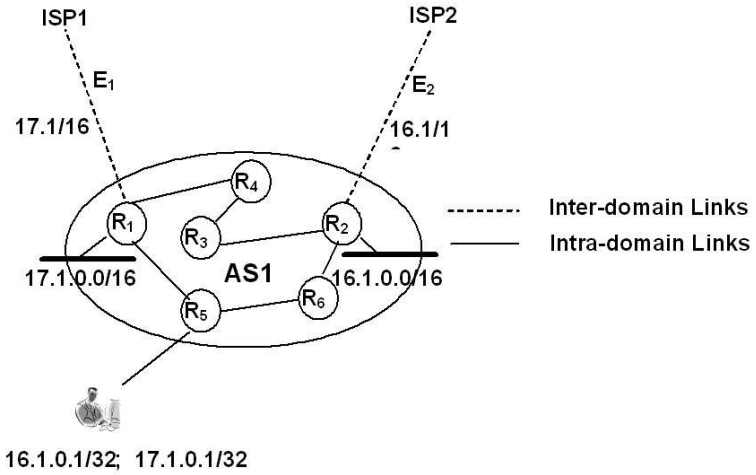


Fig. 2.5 Managing user movement in a stub AS.

2.5.3 Host Identity Protocol (HIP) Architecture

The Internet has essentially only two global namespaces (IP addresses and DNS names) to meet all its computing needs. One of the main deficiencies of the current namespaces is that it does not allow dynamic readdressing. This handicap has prevented a large number of constructive research proposals in IP multihoming and mobile computing from being successfully implemented. The Host Identity Protocol architecture ([115],[117]) has been proposed by Internet Engineering Task Force (IETF) to address this issue of dynamic readdressing as well as other deficiencies in the current namespaces. The IP addresses are currently used both for topological location (routing packets) as well as end-point identifiers (identifying the physical network interface). In the HIP architecture the locators and end-point identifiers are separated. IP addresses are still used as locators but the end-point identification is done using host identifiers. In this architecture, address changes are straightforward. A node while in communication can directly change its IP address by changing its binding between the host identifier and IP address. The node informs its peer about the address change by simply sending HIP readdress packet. Once the peer accepts the new address, the active sessions (as well as new sessions) between the peers will be seamlessly transferred to the new address.

The HIP architecture has been shown to work well between hosts that use this architecture. There has been a consistent effort from the IETF to make this proposal an Internet standard and subsequently for its Internet-wide deployment. The latest developments on this architecture are availableonline in its official charter at [116]

2.6 Related Studies

In the previous sections as well as in the following chapters we review a number of works which are related to our work in terms of the techniques used, approach and application. In this section, we study those approaches that are related to our work in terms of the input, network model and goal. In particular, we look at models that try to load balance traffic in a multihomed environment. We study the evolution of such models and their quest to represent a real multihomed environment.

Online problems are those which receive their input incrementally, one piece at a time. In response to the input seen so far, an online algorithm must generate output not knowing future input. An extensive systematic study of online algorithms started after Sleator and Tarzan ([103]) suggested comparing an online algorithm with an optimal offline algorithm. The term *competitive analysis* was coined to measure the performance of online algorithms ([75]). In a competitive analysis an online algorithm A is compared to an optimal offline algorithm OPT. An optimal offline algorithm knows the entire input sequence in advance and can process it optimally. Online algorithms have been used to solve many interesting problems including those of load balancing. In the classical online load balancing problem ([27]), there are n parallel machines that is used to process a sequence of independent tasks or jobs. Upon arrival, a task is assigned to one of the machines immediately. Arrival pattern of the tasks is not known. Each task is associated with a weight and a duration which may or may not be known in advance. The load of a machine is the sum of the weights of the jobs present on the machine at that instant. The goal is to minimise the maximum

load, though other goals can also be considered. In special cases, re-assignments of tasks to machines are allowed to improve the overall performance. However, re-assignments are an expensive process and should be limited. The online load balancing problem arises in many scenarios where allocation of resource is involved. An example (cited in [27] also) is where each machine represents a communication link with bounded bandwidth. The problem is to assign request for bandwidth from users to one of the links such that maximum load on the links is minimised. Azar et. al. ([27]) proved using randomised algorithms, that the optimal competitive ratio achievable for the load balancing problem with n machines is 2-1/n. Randomised algorithms have often been used to solve online problems since on an average they can obtain better competitive ratios. The power of randomisation to tackle online algorithms has been explored in [26].

One may consider that online load balancing models are similar to our problem in the sense that the mode of input and final goal is analogous to our problem. Moreover, it also allows re-assignment of user traffic to improve performance. However, there are significant points of differentiation with our problem mainly with reference to the assumptions of network traffic. Firstly, in the online load balancing, bandwidth of the links are assumed to be known (constant) whereas in our case it will vary because of cross traffic and route changes. Secondly, in the load balancing model, bandwidth of a traffic flow will remain constant for the entire duration of the flow. In our problem, both the bandwidths as well as duration of flows are unknown and the bandwidth of a flow will vary over time. Thirdly, all of the proposed online load balancing algorithms available in the literature have a centralised controller. In this work, our goal is to propose both centralised as well as distributed algorithms for our problem.

Realising the need to study the dynamic nature of network behaviour from a host's perspective, Karp et. al. ([76]) initiated such a study by proposing a simple model. In this

model, a host A wants to transmit the maximum possible packets (with no packet drops) to another host B. An *adversary* is used to dynamically vary the bandwidth of the link connecting the two hosts. A number of randomised algorithms were proposed for this simple model ([76], [25]). Later this simple model was extended to represent a more realistic network ([37]). We call this extended model Karp's model, so as to bring a distinction with our model. In Karp's model, host A and host B are connected by N paths. The bandwidth on each of this path varies over time. More specifically, the bandwidth, $c_i(t)$, of path i at time t is not known except for a upper and lower bound called the pinning interval. $x_i(t)$ is the amount of traffic offered to the path i; it is assumed that the $x_i(t)'s$ can be measured. The total traffic D is then given as, $D = \sum_{i=1}^{N} x_i(t)$. The goal is to distribute D over the N paths such that there is no congestion on the paths (i.e. no packets are dropped). In case a feasible solution exists, the aim is to find the solution with the least *convergence time*. The other aim is to re-assign as little traffic as possible in between the paths. In other words their goal is to develop an iterative algorithm which narrows the pinning interval during each iteration till it finally converges to c_i. The authors ([37]) model the uncertainty in the problem as a search game. They attempt to solve the search game in two steps - static and dynamic search games. Initially in the static search game, algorithms are proposed assuming both available bandwidth of the paths as well as information about the network topology are known. Later these assumptions are relaxed in the dynamic search game.

Given that the Karp's model tries to re-distribute network traffic across multiple paths, their network model is analogous to the one we consider in this dissertation. In order to highlight the complexity of our problem we initially perform an *offline analysis*, where we assume both the available bandwidth as well as input traffic to be known. The offline analysis of our problem is equivalent to the static search game of Karp's model. We demonstrate that an exact solution of the offline problem is possible only if the input traffic is appropriately discretised. Similar observations have been made in the Karp's

model too. They observed that in the absence of appropriate levels of discretisation, one may not be able to find a feasible solution. For the dynamic scenario, the aim of Karp's model is to distribute a given total traffic D over N paths in the presence of imprecise information about network topology and bandwidth. In the real world in addition to network measurements being inaccurate, the arrival pattern of input traffic will be dynamic and unpredictable. To make our model more realistic we assume that the input traffic is unknown and that it arrives in installments (as in the case of online problems). We do not make any presumptions about the network traffic. Based on the input network traffic seen so far, our aim is to make provisions for the next time period, such that the traffic load on the multiple links is balanced. Although our assumptions made about network behaviour is similar to that of the Karp's model, the mode of input traffic in our problem is comparable to that of the online problems. In other words we have incorporated the uncertainty involved in both the Karp's model as well as in online problems. This makes our problem more difficult than either of these two models.

Chapter 3

Survey of Bandwidth Estimation Techniques

3.1 Introduction

Traffic engineering techniques cannot create additional bandwidth; they give preferential treatment to some packets while restricting other packets. In a multihomed site all the links may not be fully utilised at any given instant. While one link is congested other links may be lightly loaded. In order to load balance the links, the first step would be to monitor and detect congestion on the links. In this work we propose to monitor the available bandwidth on the egress links. Our primary objective is to fully utilise the resources at our disposal. We understand there exists a common bottleneck point in the provider's network through which all flows of the end network has to pass through. We call such a path as the so called *connecting path*. The main congestion is assumed to be on this path. A formal definition of *connecting path* is given in chapter 4. In this chapter our aim is to devise a methodology for measuring the available bandwidth of the *connecting path*. There are two key technical issues underlying the design of such a tool. Firstly, the tool should be able to identify the common bottleneck point. Secondly it should provide an

estimate of the available bandwidth of this bottleneck path. In general, the tool that we seek to design should be able to measure hop-by-hop available bandwidth of Internet paths and locate bottleneck positions in the Internet. Additionally, the tool should perform the measurements without cooperation from the remote end. The Internet being a collection of uncooperative networks, this is an important requirement from a deployment point of view. Finally the tool should be light-weight.

The ability to locate bottleneck link along Internet paths, other than allowing multi-homed networks to avoid bottleneck links is of great interest to end users as well as network operators. End users can use the information to estimate performance of a network path to a given destination. The knowledge of bottlenecks can provide network operators insight into the causes of congestion and ways of circumventing it. Flows that share a common bottleneck resource can benefit from cooperative congestion control strategies ([95]). Due to the intriguing nature of the problem and its practical importance, a number of techniques as well as tools have been proposed to measure bandwidth on Internet paths. Given the large array of tools and techniques publicly available to estimate bandwidth, we first review these open source tools. Our aim is to inspect the possibility of tailoring one or more of these tools to meet our prescribed design. The survey is in two parts. Initially we examine those tools which can measure available bandwidth on Internet paths. The intention of this first part of our evaluation is to review the existing technologies used to estimate available bandwidth and comment on their accuracy. In the second part we analyse those tools that can locate bottleneck points on Internet paths.

3.2 Bandwidth Estimation Techniques

In the context of data networks, the term capacity or bandwidth of a path means the maximum bandwidth that a flow can achieve when no other traffic is present. Available bandwidth means the maximum bandwidth that such a path can provide to flows,

given the existing traffic it is already carrying. Network measurement techniques are of two categories: passive measurement ([61]) and active measurement ([34], [63]). Passive measurements work on network traces collected earlier. Although they are efficient and accurate, their scope is limited to network paths that have recently carried user traffic. In this work we concentrate on the active probing techniques only. Active probing techniques on the other hand are more useful since they can be used to determine the instantaneous bandwidth. Typical methods of active probing scheme are packet pair/train dispersion (PPTD), variable packet size (VPS) probing, Self-Loading Periodic streams (SLoPS) and Trains of Packet Pairs (TOPP).

In packet pair probing ([34], [36]), the source sends multiple packet pairs to the receiver. From the measure of the dispersion experienced by the packet pairs, the receiver computes the end-to-end capacity. In order to cancel the influence of cross-traffic on the measurement of capacity a number of packet-pairs are sent and statistical methods are used to filter out erroneous measurements. In VPS probing ([69], [31]), multiple packets of a given size are sent. This technique uses the TTL field of the IP header to force the probing packets to expire at a particular hop. The source uses the ICMP error messages received from the routers to measure the RTT to that hop. TTLs of the probe packets can be suitably designed such that the TTL of a pair of probe packets expire at each hop. Thus VPS probing can be used to measure the capacity of each hop along a path. In SLoPS ([70]), a series of equal-sized packet probe trains is sent at a particular rate. Depending on the trend of one-way delays experienced by the stream, the sender varies its sending rate and attempts to bring the stream rate close to the available bandwidth. If the streaming rate R is greater than the path's available bandwidth, the stream will cause a short term overload in the queue of the bottleneck link increasing the one way delays of the probing packets. On the other hand, if the streaming rate is lower than the available bandwidth, the one way delays of the probing packets will not increase. While SLoPS overcomes the

inaccuracy in existing probing techniques, it requires a large number of packet streams and a very long measurement time which makes it unsuitable for real-time applications. TOPP ([84]) like SLoPS, sends packet streams and gradually increases the stream rate to measure the available bandwidth. The difference between the two methods is in the statistical processing of the measurements. More details on bandwidth estimation techniques and a taxonomy of public available bandwidth estimation tools can be found in the survey paper by Prasad et. al. ([91]). An up-to-date list of tools for measuring Internet performance is maintained online at [16].

Bandwidth estimation tools can be classified as either a double end-host (DE) or a single end-host (SE) tool ([114]). DE tools must be installed in both the source and destination hosts of target paths, while SE tools in only the source hosts. Since DE tools have access to both ends they are generally more accurate than SE tools, but they are less scalable since cooperation of both ends is required. Our main aim being to find tools that can be deployed in an uncooperative environment, in table 3.1 we provide a list of SE tools along with their measurement metric and methodology used. Measuring available bandwidth is complex in practice since it depends on many other factors including latency, loss rate, network path load and TCP implementation details. A number of studies have shown that existing techniques to measure available bandwidth provide only a rough estimate. Estimating available bandwidth requires more coordination between the two ends. This is evident from the limited number of SE tools that are present to measure available bandwidth. Zhow et. al. ([114]) presents a comprehensive analysis of the difficulties faced by DE as well as SE tools in estimating available bandwidth.

3.3 Locating Bottleneck Links

In the preceding section, we have examined a number of techniques and tools that can measure the bandwidth of bottleneck links. Our next objective towards measuring the

Table 3.1 Bandwidth Estimation Tools with Single-ended Control.

Tool	Measurement Metric	Methodology
bing	End-to-end capacity	PPTD
bprobe	End-to-end capacity	Packet Pair
cprobe	End-to-end available bandwidth	Packet Pair
pipechar	End-to-end available bandwidth	Packet Pair
clink	Per-hop capacity	VPS
nettimer	End-to-end capacity	Packet pair
pathchar	Per-hop capacity	VPS
pchar	Per-hop capacity, Latency and Loss	VPS
sprobe	End-to-end capacity	packet pairs
treno	Bulk transfer capacity	Emulated TCP throughput

available bandwidth of the bottleneck path is to find the location of the bottleneck point. Intuitively, if we can locate the bottleneck point, then we can measure the available bandwidth using one of the tools reviewed in the previous section. In this section, we review four tools that can locate bottleneck points along Internet paths - BFind, Spatio-temporal available bandwidth (STAB), Pathneck and BNeck.

BFind ([22]) operates in a single-ended mode. First, BFind obtains the propagation delay of each hop to the destination. For each hop along the path, the minimum of the measured delays is used as an estimate for the propagation delay on the hop. The minimum is taken over delay samples from 5 traceroutes ([15]). After this step, BFind starts a UDP stream to the destination at a low sending rate and monitors the delay. It also starts concurrently a variant of traceroute. The hop-by-hop delays measured are combined with the raw propagation delay computed initially to obtain rough estimates of the queue lengths on the path. BFind gradually increases the sending rate to build up the queues of intermediate routers. BFind then finds the routers whose queue sizes have increased and the corresponding link is identified as the bottleneck link. The lowest probe rate that induces build up of the queues is regarded as the available bandwidth. The main drawback of BFind is that it needs to send a large amount of data to fill up the links.

STAB ([94]) is a tool that can identify the link with the least available bandwidth on an end-to-end path. The tool introduces several novel concepts like *self induced congestion*, *packet tailgating* and *chirp trains*. Unlike, BFind it is a relatively light-weight tool. However, its main drawback is that it requires access at both end points of the path.

Pathneck ([62]) is an active probing tool based on a novel probing technique called Recursive Packet Train (RPT) which allows end users to locate bottleneck links on the Internet. A choke link is defined as any link that has a lower available bandwidth than the partial path from the source to that link. The upstream router for the choke link is called the choke point or choke router which corresponds to our definition of congestion point (X) as defined in chapter 4 (section 4.3.1). Figure 3.1 shows an example of a RPT. The idea is to combine measurement and load packets in a single probing packet train. Each box is a UDP packet and the number in the box is its TTL value. Measurement packets are 60 byte UDP packets with properly filled-in payload fields. Load packets are 500 byte UDP packets used to emulate the behaviour of regular data traffic. The probing source sends the RPT packets in a back-to-back fashion. When they arrive at the first router, the first and last measurement packets of the train expire, since their TTL values were set to 1. As a result, the packets are dropped and the router sends two ICMP packets back to the source. The time difference between the arrivals of the two ICMP packets is called the packet gap. Due to the way TTL values are set in the RPT, the above process is repeated at each subsequent router. The sequence of packet gap values from routers along a path is called the gap sequence. The core part of the algorithm for calculating candidate choke points involves matching the gap sequence to a step function, where each step corresponds to a candidate choke point. For each path, the choke points are ranked based on their average gap value in the probing set. Pathneck does not measure available bandwidth of the bottleneck path. However, based on the average per-hop gap values, it provides a upper

or lower bound for the available bandwidth of each link.

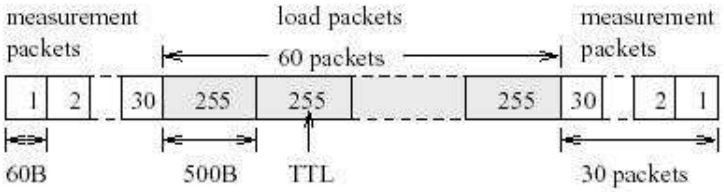


Fig. 3.1 Recursive Packet Train (RPT).

Alluding to the usefulness of their tool the authors, Ningning Hu et.al claim that pathneck can be used to address the following two questions in a multihomed network - (i) Given a set of popular destinations, which upstream provider should be considered, and (ii) Given a set of upstream providers, which provider should be used to reach a given destination? In actual networks, one would probe from the same source to several destinations through different upstream providers to measure available bandwidth on the uplink paths. In the experiment carried out by Ningning Hu et.al, the probing nodes were not located in the same city. They, therefore, simulate multihomed networks by choosing several nodes and grouping these nodes based on their geographic proximity. For each geographic group, bounds on the available bandwidth of the paths from each member in the group to the same destination are measured. If the lower or upper bound on available bandwidth from the worst path compared with any other path in the group is more than 50 percent, multihoming is declared to be useful. Results show that more then 70 percent of the cases are useful ones.

BNeck ([113]) is an active probing tool that tries to combine the functionality of locating the bottleneck link, measuring link capacity and available bandwidth in a single tool. The design of its packet stream is similar to that of pathneck. The TTL of the packets in the stream increment linearly from both ends but all packets are of the same size. The source

estimates the dispersion of the train by measuring the time gap between the two ICMP packets from each router. BNeck once again uses the idea proposed in pathneck to locate the bottleneck points. However, to measure the capacity and available bandwidth of the links, additional probe packets of different sizes are send by the source. This makes the tool relatively more heavy weight than pathneck.

3.4 Properties of single-ended tools

Based on our evaluation of a large number of publicly available tools to locate bottleneck points and measure its available bandwidth, we find that the tool pathneck nearly fulfills all our requirement. Pathneck has a single-ended control, it is light-weight and can detect bottleneck links on end-to-end path. The only objective that pathneck does not fully meet is that it does not measure the available bandwidth of the bottleneck link but provides a upper or lower bound. Given that the bottleneck point can be identified using pathneck, we can use tools like cprobe or pipechar to estimate the available bandwidth of the bottleneck path. This would, however, mean generating additional probe packets to measure available bandwidth (similar to the idea used in BNeck). Besides, Dovrolis et. al. have demonstrated that what cprobe and pipechar estimate is not the available bandwidth but asymptotic dispersion rate, a quantity which is related to available bandwidth but is not the same ([34]). In view of the above facts, the available bandwidth information provided by pathneck is the cheapest to obtain and is considered good enough.

From our analysis of pathneck, it is evident that the technique employed by the authors is a variant of the variable packet size probing (VPS) bandwidth estimation technique, described above in section 3.2. The measured RTT to each hop consists of three delay components in the forward and reverse paths: serialization (transmission) delays, propagation delays and queuing delays. Thus single end-host tools like pathneck, although primarily want to measure the metrics of the forward path, they unavoidably conflate the

delays of both the forward and reverse paths ([114]). While this property of conflating the delays of the reverse path may be a cause of concern for others, in the context of our problem, it works to our favour since we want to basically measure the metrics of incoming traffic. Assuming there is very little traffic on the forward path, the metrics measured by pathneck primarily reflect the properties of the backward path. Our validation of pathneck on actual Internet paths in an access network reveals that the delays reported by pathneck are indeed for the reverse path, destination to source.

Estimates of single-ended tools like pathneck which are largely built on "echo" techniques rely on the timing of the ICMP replies from the target. However, many routers on the Internet ignore ICMP and do not send back ICMP error replies. Tools based on ICMP can also yield significant capacity underestimation errors if the measured path includes store-and-forward layer 2 switches which introduce serialization delays but do not generate ICMP replies because they are not visible at the IP layer.

3.5 Internet validation of pathneck

In this section we present the results of validating pathneck on Internet paths. As mentioned in the documentation of the tool, the number of load packets used is critical to its performance. Currently, the number of load packets required for a link is empirical and needs to be discovered using trial and error. In short pathneck does not accurately measure available bandwidth and in certain scenarios it may fail to provide results. To check whether the results provided by pathneck are acceptable, we need to know the instantaneous available bandwidth of all the links on the path. However, it is not feasible to obtain this information for all hops on Internet paths. We, therefore, devise a scheme to provide a check on the results reported by pathneck. We SNMP probe the edge router and measure the incoming octets in a given time period. This value divided by the time period yield the incoming bandwidth utilisation of the particular link. The difference of

the capacity and utilisation of the link is assumed to be the available bandwidth of the incoming path. Strictly speaking, the metric so computed is the available bandwidth of the link that connects the end network to that of the provider's network. Assuming that there is only one path that connects the end network to the provider's network, this link will always be the first hop of any choke point reported by pathneck. Thus our computed value of available bandwidth will be an upper bound on the available bandwidth reported by pathneck.

The tool was tested on different Internet paths. In tables 3.2, 3.3 and 3.4 we show the comparison of the output of pathneck with our measured upper bound of available bandwidth on three Internet paths provided by three different ISPs. All bandwidth measurements are in Mbps; "ub", "lb", and "uk" stands for upper bound, lower bound and unknown respectively. If the available bandwidth reported by pathneck is less than our computed upper bound, the output of pathneck is accepted else discarded. We found that under high traffic scenario, pathneck fails to detect choke points. In such cases, the value of the upper bound is considered as the available bandwidth of the bottleneck path.

Output of SNMP Probe (Upper Bound): 1.449				
Output of Pathneck				
Hop Count	RTT	Choke Point	Estimated Bandwidth	Hop Hostname
01	18.222	.	1.149 (lb)	static115-2.staticcal.vsnl.net.in
02	312.664	.	588.674 (lb)	static191-11.staticcal.vsnl.net.in
03	34.126	1	*1.056* (ub)	59.163.16.73.static.vsnl.net.in
04	0.000	.	0.000 (uk)	0.0.0.0
05	78.743	2	1.038 (ub)	59.163.16.13.static.vsnl.net.in
Final Output (Available Bandwidth): 1.056				
Internet Service Provider: VSNL (Capacity: 2 Mbps)				

Table 3.2 Detecting bottleneck links and measuring its available bandwidth on a low bandwidth Internet path.

Output of SNMP Probe (Upper Bound): 1.16				
Output of Pathneck				
Hop Count	RTT	Choke Point	Estimated Bandwidth	Hop Hostname
01	39.944	1	1.447 (ub)	kol-guwahati-backbone.ernet.in
02	39.678	.	1.447 (lb)	202.141.139.4
03	99.008	.	1.169 (lb)	125.20.0.253
04	118.277	2	0.000 (uk)	125.21.167.25
Final Output (Available Bandwidth): 1.16 Mbps				
Internet Service Provider: ERNET (Capacity: 2Mbps)				

Table 3.3 Detecting bottleneck links and measuring its available bandwidth on a low bandwidth Internet path.

Output of SNMP Probe (Upper Bound): 31.60				
Output of Pathneck				
Hop Count	RTT	Choke Point	Estimated Bandwidth	Hop Hostname
01	0.492	.	34.422 (ub)	210.212.8.57
02	110.553	.	12905.551 (lb)	210.212.8.50
03	15.616	2	21.954 (ub)	218.248.255.10
04	58.382	3	17.880 (ub)	220.227.53.238
05	58.371	1	*11.547* (ub)	220.224.140.33
Final Output (Available Bandwidth): 11.547				
Internet Service Provider: BSNL (Capacity: 34 Mbps)				

Table 3.4 Detecting bottleneck links and measuring its available bandwidth on a high bandwidth Internet path.

Chapter 4

Threshold Based Approach to Manage Incoming Traffic

4.1 Introduction

Even though there is talk of cheap bandwidth and easy availability of bandwidth, proper management of the available bandwidth will always be required as costs, quality of service, and even insufficient bandwidth have to be addressed. In chapter 1, we discussed about the increasing popularity of multihomed techniques to optimise network resources and improve download speeds apart from providing resilience from link failures. Multihoming has been commonly associated with large routers and complex BGP configurations ([2], [109], [20], [83]). However, the use of BGP in multihoming is not the only solution. Today, multihomed networks employ a variety of route control techniques other than solutions along the BGP line ([8], [21], [57]). In this chapter, our goal is to demonstrate how the practical benefits of load balancing can be reaped in an actual multihomed environment. The networks that we focus on in this work are *small* end networks spread over a limited geographical area. Such networks generally have high bandwidth links to connect the nodes of the network. We concentrate both on improving the Internet performance as well as optimising the use

of network resources.

More specifically, in this chapter we present a scheme to utilise the available bandwidth of egress links in a multihomed local area network (LAN). A threshold limit is defined for each of the egress links to infer congestion. The scheme monitors the utilisation of the egress links and re-assigns users or groups of users from one link to another as congestion is detected on a link. A centralised bandwidth manager acts as the default gateway for all the participating nodes. This central system is entirely responsible for making and implementing the decisions. Network Address Translations (NAT) is used to provide transparency of route changes to the source nodes. A general technique of load balancing in multihoming networks is to manually assign the user traffic to egress nodes based on past performance (we call this the *default* approach). We compare our load balancing algorithm to the *default* approach and evaluate the improvements in Internet round-trip times (RTTs). We quantify the effectiveness of our load balancing approach by measuring the number of user re-assignments. We also address a number of other practical issues such as usefulness of past history to define link thresholds, admission control and clustering of users to make management of routing tables easier.

To demonstrate the effectiveness of our scheme, we implement a prototype of our scheme in an actual campus network. In the first part of our experiment we highlight the benefits of multihoming vis-a-vis singlehoming. The performance gains of our multihoming route control technique are comparable to existing multihomed techniques, but our approach is much simpler and practical. Existing multihomed techniques rely on choosing the best ISP link ([21]) to realise performance benefits whereas our aim is to improve the performance by optimising the utilisation of the multiple egress links at our disposal. In order to evaluate our approach for the general case, where an end network can be multihomed to K ISPs, we use a 3-multihomed end network. In the later part of our evaluation, we show the benefits

of multihoming using a coarse granularity of load balancing. Given a network with a fixed set of proxy servers used to access the Internet, we show that without disturbing the existing network setup our route control strategy can be transparently applied for performance benefits. We identify tactics to fine-tune the value of link thresholds. We illustrate that by carefully choosing the threshold limits an overall improvement in performance can be achieved even with a coarse granularity of load balancing. In this chapter our main focus is on the practical aspects of load balancing in multihomed networks. A detailed analysis of load balancing in the presence of imprecise network information and alternative tactics to overcome the short-comings of the threshold approach are given in the following chapters.

Evaluation of existing bandwidth monitoring tools and our strategy to monitor the egress links has already been discussed in chapter 3. In section 4.2, we give a formal statement of the load balancing problem. We describe our solution mechanism in section 4.3. Section 4.4 deliberates on a number of additional design issues that need to be considered during actual deployment. We give details of our implementation in section 4.5. The experimental setup and results are presented in section 4.6.

4.2 Problem Statement

Figure 4.1 shows an enterprise network (LAN) connected to two ISPs. Given a destination D, an ideal multihoming load balancing algorithm would select an external link such that latency to access the site D is the least through that link. However, in practice always selecting a link that has the least latency would require that we probe all the egress links for every given destination. This is a practically infeasible proposition. The research issue addressed is to make the best use of incoming bandwidth available from the different ISPs. The incoming traffic on the different lines is sought to be controlled by appropriately distributing the corresponding outgoing traffic among the K ISPs. While load balancing the egress links, we additionally need to consider the following:

- not to allow average round-trip times (RTTs) to deteriorate.
- not to cause disruptions in user connections; if traffic from a node is re-routed from one connection to another, ongoing transfers are to continue unaffected.
- to provide different quality of service to different users.

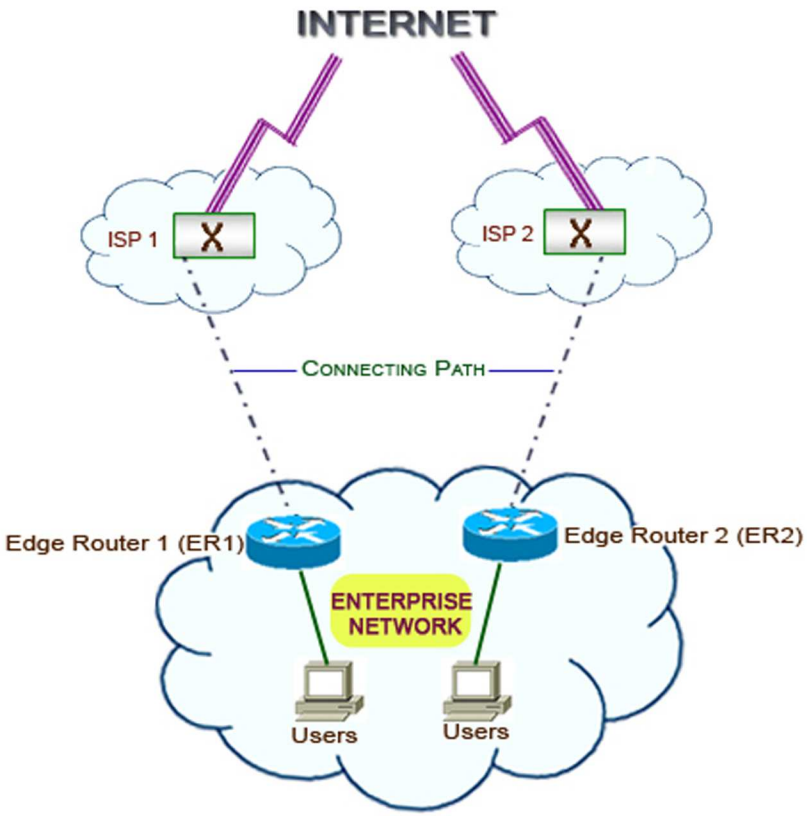


Fig. 4.1 A typical enterprise network connected to two ISPs. The user traffic is assigned to the two ISPs using static routing.

The key assumption made in this chapter is that both the outgoing and incoming traffic of a user follow the same egress link. The other assumptions are:

- Traffic is primarily TCP. All connections are initiated from within the enterprise network. Since our focus is on access networks this assumption is not far-fetched.

Services offered by the enterprise like web servers can be assigned static public IP addresses and placed outside the realm of load balancing.

- The main congestion is assumed to be on the incoming bandwidth only. Thus we basically need to monitor the incoming traffic.
- Sufficient bandwidth is available within the network and there is no congestion within the network. Latency experienced by Internet traffic is only due to congestion on the paths connecting the network to the Internet.

4.3 Solution Details

The solution proposed for controlling the incoming traffic consists of three basic steps: (i) monitor the bandwidth of all the egress links, (ii) based on the results of monitoring, if the link utilisations are above a pre-defined threshold, re-assign the nodes to a less congested link and (iii) finally direct the traffic from nodes to their assigned egress link.

4.3.1 Monitoring Incoming Bandwidth

Correctly monitoring utilisation on the egress links is crucial to the performance of our algorithm. The first issue that needs to be addressed in the monitoring of egress links is, what parameters do we monitor to detect congestion? Our load balancing algorithm needs to ensure that the average response times do not deteriorate. A simple and straight forward approach would be to monitor round-trip times on the egress links. However, measurement studies have shown that there is little correlation between increased delays (or RTTs) and congestion ([68], [100]). In bandwidth management a metric of great significance is available bandwidth. By available bandwidth we mean the maximum bandwidth that a path can provide to flows, given the existing traffic it is already carrying. The performance of a link also depends on several other Quality of Service (QoS) parameters like delay, jitter etc. Bandwidth being one of the most prominent QoS parameter, we propose to monitor the

(incoming) available bandwidth of the external links. Moreover, if a link has the necessary bandwidth than the other QoS parameters such as delay and jitter will normally become minimised.

To measure available bandwidth, we need to know the two ends of a communication path between which we will measure this metric. One end of the path would obviously be the egress router of the end network, but what will be the other end? To optimise the Internet experience of end users, it is crucial to understand the location and traffic load of bottleneck links. Most of the traffic engineering techniques which target multi-homed networks usually monitor end-to-end Internet paths ([21], [57]). In such end-to-end monitoring, ideally one would like to monitor the path for all possible destinations and select the one with least latency. However, even a small network will access thousands of destinations and monitoring the path for all these destinations is practically infeasible. A popular solution is to select the most popular destinations, but even this can be infeasible for large networks. Moreover, how to categorise a destination as popular is contentious. The challenge with such selective monitoring is to avoid biasness due to a narrow view of the network from few probe sites.

In chapter 1, we studied that Internet download speeds may be affected either due to *local* or *remote* factors. Performance considerations due to *remote* factors require cooperation from external agencies, but those due to *local* factors can be managed single-handedly by the end network. In this work we do not want the Internet performance to be affected by *local* factors to the extent possible. In the context of Internet bandwidth, this would mean we want to ensure that the bandwidth available to the end network is fully utilised. We assume that there is a point "X" (as shown in figure 4.1) in the provider's network to which all packets from the end network have to travel to, and that beyond the point X, there is sufficient bandwidth available (highways). The congestion (or restriction in available

bandwidth) is assumed to be only up to point X. In fact there may be more than one path to the point X, but typically an ISP will provide one path through which all packets of a customer are routed. The path to this point X from the egress router is referred to as the *connecting path*. In the simplest case, the path could be of length 1 if the provider's peering border router is point X. In this case where the path is one hop, we can directly probe the edge router and find out the bandwidth usage. For the general case the point X will not be known and has to be discovered. The challenge is to have a scheme that is light-weight and does not rely on co-operation from nodes at the remote end of the connecting path for making an estimate. We adapt an existing tool pathneck ([62]) to infer the congestion point and measure the available bandwidth of the *connecting path*. In the preceding chapter we provided a comprehensive analysis of the various tools and techniques available to measure bandwidth and locate bottleneck links.

4.3.2 Assigning Traffic to Egress Links

The second step is to assign network traffic among the various providers. The basic granularity of assignment in our load balancing approach is a user (IP address). Initially users are assigned links as per the default static routing policy of the organization. A threshold value is defined for each of the links. The threshold value of a link indicates the maximum available bandwidth of the link. Based on the monitoring of the *connecting paths*, if the utilisation of a link exceed its predefined threshold, than a user is moved from the congested link to the least under-utilised link. The users are moved one at a time from the congested link. The process of moving users continues until the utilisation of the link goes below threshold, or there are no links available to move the users. The users are moved back to their originally assigned links when the utilisation of the link goes below the threshold. The reason behind moving back a user to its default link is that we do not want to disturb the long term goals of the network. In networks where static routing is used to distribute user traffic, network administrators apportion the traffic on the basis of past performance of the

ISP, link capacity and user traffic profile. Our load balancing scheme does not intend to interfere with the long term strategy of the network. Our aim is to manipulate the routing strategy for short terms in order to optimise the bandwidth usage. The other reason is that the count of the number of times a user switches from its original assigned link to a different link and moves back provides a measure of the effectiveness of our route control strategy. Consider a scenario where we have two links. If the static route assignments are such that the first link is always over-utilised and the second link under-utilised, then moving a few users from the first link to the second will result in overall improvement of performance and at the same time the number of re-assignments will be low. Therefore, this count provides us a check on the biasness of the static routing policy. When our load balancing algorithm is run for a sufficiently long period, the number of times a user moves out and returns to its original link should be ideally equal. In case there are more than two egress links, a user may get moved from its default link to a second link and then to a third link and so on. However, if we track only the two situations where a user moves out and moves back to its default link than their count should be ideally equal.

4.3.3 Addressing Issues

A stub network may have an independent IP address block of its own or it may use the IP addresses assigned by its providers. Assume the enterprise network has been allocated a class B IP address range. To switch traffic from one line to another, the enterprise will announce a subnet of this address to each of the egress links. In order to route traffic through a particular link, a user will use an IP address advertised on that link or use NAT to translate its address accordingly. More likely, an ISP will give to the enterprise network a set of IP addresses to use and it will only route traffic with these IP addresses as source/destinations. In such a case, the only way for a node to directly access the Internet through an egress link is to be allotted an IP address of that link. If a node is to connect to all the links, than it will be have to be given an IP address corresponding to each link. It

is unlikely that the number of IP addresses available will allow such a scheme to be implemented. One may argue that a node will need only one IP address at a time, and therefore a manager could allocate IP addresses on demand. However, as we shall see, a node may need to be communicating through more than one link at a time to avoid disruptions in TCP connections. This will require a node to handle connections using more than one IP address and it won't be possible to handle this transparently without making changes to the node software. Proposals to modify the client network stack in general will not be acceptable.

The network model that we propose to use is the NAT-based model, details of which have been given earlier in chapter 2. NAT is used to redirect traffic to the desired ISP link. This approach is transparent to the nodes. While NAT is not able to handle all network traffic, it is commonly used to operate with a limited number of IP addresses and traffic that NAT cannot handle can be treated specially. A central bandwidth manager acts as a central router to the enterprise network and all nodes treat the bandwidth manager as their default gateway. The manager in turn uses NAT to translate the source IP addresses as desired and forward packets to the appropriate router and manages the active connections of a node while switching the node from one egress link to other. In the remaining part of the thesis we use the term router to mean this bandwidth manager.

4.4 Additional Design Issues

4.4.1 User Classes

In order to make the management of users and routing tables simpler we group users with similar priority into user classes. However, selecting the maximum size of a user class is not trivial. If the size is very small it will lead to a finer granularity of load balancing; but then, the size being small the traffic generated by the class is likely to be less. Thus

during congestion we may need to move more number of users, to bring down the link utilisation below its threshold. In other words our load balancing algorithm is likely to take more time to converge. The number of entries in the routing table will also become large and unmanageable for a network with a huge user population. On the other hand if the user class size is big, it will lead to a small, efficient routing table. But a large size may cause users to oscillate between links. Intuitively, if the user class size is very large then switching the user to a second link will make the first link under-utilised and the second link over-utilised. This will prompt our routing scheme to move back the user to the first link or to a different under-utilised link in case there are more than two egress links. Such user oscillations (a so called *ping-pong* effect) can adversely affect the performance of our route control technique. The other problem with a large user class size is that it will lower the performance benefits due to the coarse granularity. This leads to the question, what is the optimal size of a user class? The size of a user class will depend on the Internet usage intensity of the users. Theoretically, a single long-lived TCP flow can fill up the entire bandwidth of a link. In chapter 5, we show using simulation, that even the switching of a single user can trigger the *ping pong* effect. Ideally the size of a user class should be dynamic and its size should depend on the usage intensity of the users.

A practical way to address this issue of ping-pong, is to define two thresholds for each link - a upper threshold and a lower threshold. When a link exceeds its upper threshold then it signals congestion on that link and if it is below its lower threshold then it means there is bandwidth available on the link and a user can be accommodated. If the utilisation of a link is between its upper and lower thresholds, then it means that the link is in a steady state. No user will be moved from the link but it will not accommodate any new users either. The other approach is to empirically determine the user class size from experiments. In chapter 5 we tackle the issue of user oscillating between links by ensuring that once a user has moved, it is not allowed to be re-assigned for the next few iterations.

4.4.2 Defining Thresholds

A correct definition of link thresholds is central to the performance of our algorithm. By threshold, we mean the maximum traffic a link can carry without affecting performance. In the simplest case, we can measure the maximum traffic that a link can carry without dropping of packets and fix this value as the threshold. Even if a fixed bandwidth is leased from an ISP, the lease may be in a shared mode, or the ISP may not be able to provide guarantees upstream. The maximum traffic carrying capability of a link will, therefore, be less than its capacity. Traffic carrying capacity of a link will depend on the cross traffic present at that instant. In general the threshold of a link will vary depending on the type of service agreement with the provider, behavior of users and time of day. A fixed threshold value will not be able to take into account such traffic dynamics. A second approach to define link thresholds would be to correlate the thresholds with the link latencies or RTTs. As the link latency increases, decrease the threshold value and vice-versa. The issue here would be the quantum by which we will increase or decrease the threshold value. Moreover, at the same level of utilisation, latency of a link may increase or decrease. As we have stated earlier, a number of studies have shown that there is little correlation between increased delays and congestion. The increase in latency may be due to bottleneck on the remote destination and not because of cross traffic.

In figure 4.2, we plot the link latencies as a function of its utilisation. The link latencies remain low for small link utilisations but beyond a certain limit the latencies tend to increase exponentially. This is due to the fact that as the utilisation of a link exceeds its available bandwidth, packets will get dropped and re-transmissions will occur. For this particular link, the saturation point is at about 90 percent of the link capacity. The other observation is that as utilisation of a link increases, though there are no packet drops, its latency also increases. This may be due to queuing of packets in the upstream router's buffer. In general link latencies grow as utilisation of the link increases. This indicates that

in order to bring about an overall improvement in performance the threshold values should be defined in such a way that utilisation of all the links are minimised. In other words traffic on the links should be load balanced. Our third approach of defining thresholds is based on the past utilisation of a link. In figure 4.3, we show a typical hour-wise utilisation of a link. During certain periods we find that the link utilisation is at its peak and there are intervals when the utilisation almost touches zero. Based on the link history we define the threshold values, such that traffic will be moved out from the link when its utilisation is high and the link accommodates traffic of other links during periods when its utilisation is low. The possible threshold values are shown. The assumption here is that utilisation of a link does not vary to a great extent when considered period-wise. In this particular example (figure 4.3) we assume that in a 24-hour cycle, utilisation of a link at noon today will be comparable to the utilisation of the link at the same time yesterday. A similar approach would be to consider utilisation of all the links and set the threshold values proportionately to their available bandwidth. During actual implementation (in section 4.5) we show that the threshold values can be further refined empirically depending on the outcome of our load balancing algorithm.

4.4.3 ISP Pricing Models

In order to allow flexibility to customers, ISPs have different pricing structures. The pricing models may be flat-rate, usage-based, advertising-based or a combination of these models. Goldenverg et. al. ([56]) proposes algorithms to improve performance while at the same time minimising cost for multihomed networks. In usage-based models, the cost incurred is based on the volume of traffic which may be either percentile-based charging or total-volume based charging. Given the cost function of an ISP, the aim is to minimise the total cost. In [56], the ideal volume of traffic that should be assigned to each ISP is referred to as the *charging volume*. Suppose there are 4 ISPs and each of them charge based on the 95th-percentile; then the minimum charging volume of each ISP is equal to the 80th-percentile of

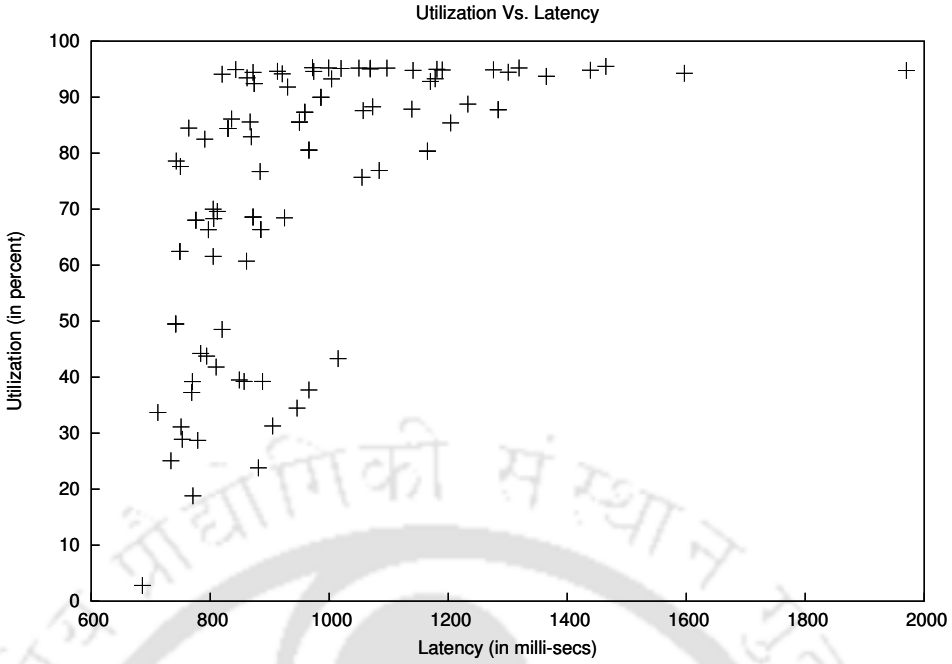


Fig. 4.2 Scattered graph of link utilisation as a function of latencies.

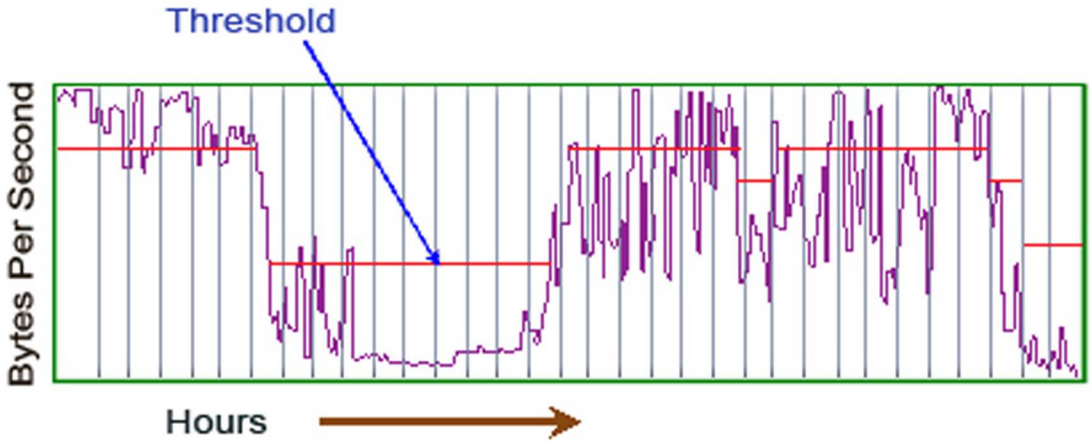


Fig. 4.3 Defining link threshold from history.

the total traffic (since 1-4*(1-95%)=0.80). Once we know the total and per-link traffic volume, the goal is to assign traffic to each ISP such that their charging volume is not exceeded.

In this work, we assume that the ISP pricing model is flat-rate based. In such a model the fee paid to an ISP does not depend on the volume of usage, hence our aim has been

to utilise the maximum bandwidth available. In case a network is connected to differential cost ISP links, then we can optimise the cost by defining the link thresholds based on the *charging volume* instead of link utilisation.

4.4.4 Handling Active Sessions

A major drawback of switching a user from one connection to another is that all the active connections of the user will fail due to the change in the source address. Most of the works on load balancing ignore such active connections with the assumption that Internet connections are short-lived. While it is true that most of the Internet flows are short-lived, majority of the packets and bytes belong to long-lived flows, and this property persists across several levels of aggregation ([99],[33]). Thus ignoring active connections may adversely affect the average throughput.

In order to preserve the active connections of a user, we allow the established connections of the user to continue undisturbed on the current link while all new connections initiated by the user are routed through the second chosen ISP link. This is done by identifying all active connections of the user and flagging them. Packets of the flagged connections are then routed by the router through the old link and then the route of the user is changed to the new link. We have to take care of multiple switches where a long lived connection in link 1 is still alive, while the user has been switched to link 2 and then to link 3. At this point of time, the user has connections on link 1, link 2 and link 3. Since we have a centralised router, handling such cases is not a problem. However, some of the new (unmarked) connections of the user directed over the new link may be part of a previous active session continuing on the old link. This could cause the session continuing on the first link to fail. Identifying all the connections of a session is very difficult. One possibility is to use a protocol specific approach to handle such active sessions. For example in the HTTP protocol, we can use session identifiers stored in cookies to identify all the connections that

belong to a session. This, however, will require that we probe packets at the application layer. In this work we assume that no new connections are initiated by an active session once the user is moved to another link.

4.4.5 Quality of Service

In this work we do not provide any bandwidth guarantees to the users. The only guarantee we try to provide is that, the congestion of the default link of each user, will be below our defined thresholds. The quality of service we can provide is in terms of the user movements. We select the user with the lowest priority as the candidate to move when congestion is detected. This will ensure that only the low priority users will face possible disruptions due to movement. An alternate way of looking at it would be to select the user with the highest priority as the candidate for movement. This will ensure that the highest priority user is moved from a congested link to another link which is not congested. However, the second link may be an inferior link, in terms of QoS parameters; hence we will not be able to ensure the same QoS to the user. This can be handled by storing the QoS parameters of each link on a look-up table. We will move the highest priority user if the QoS parameters of the second link is comparable to that of the first link, otherwise we move the lowest priority user. This, however, will require that we consult the lookup table every time we need to move a user. In our experiments, we identify some users as high priority users. These users are not considered for movement when congestion is detected on their default egress link. Rest of the users are candidates that can be selected for movement. This scheme ensures that the quality of service of these groups of high priority users remains stable.

4.4.6 Admission Control

During peak traffic hours all the egress links are likely to be congested. Thus the utilisation of all the links will be above our defined thresholds. In such a scenario load balancing techniques will not prove to be effective, since there is no bandwidth available to move

the excess traffic. What we require is some means to check the flow of incoming traffic. In such situations, a pro-active way of managing the queue at the bottleneck router like Random Early Detection ([51]) can be used to shape the inbound traffic. However, since the packets have arrived and consumed the WAN link bandwidth, shaping the traffic is not worth while. A more effective way of restricting the throughput of incoming traffic will be outgoing admission control. For instance, we may disallow users from initiating request for new downloads. Internet traffic being predominantly TCP we can use standard TCP traffic control approaches link delaying outgoing acknowledgments (details were discussed in chapter 2) will be more effective in slowing the remote destinations. However, the integration of such TCP traffic control approaches with our route control scheme is beyond the scope of the thesis.

4.5 Implementation Details

We have implemented our threshold-based route control approach in Linux. The Linux machine acts as a centralised bandwidth manager (router). The computer used is a Pentium IV machine with 512MB of memory and a number of network interface cards. All nodes of the network that participate in the load balancing scheme set their default route to the router. The router has access to all the egress links of the multihomed network. The following components were used in the implementation (i) *ip rule*, a tool to manipulate routing policy database in Linux (ii) *iptables*, a user space application program to implement NAT and IP filters (iii) a SNMP ([66]) package to probe the egress routers. SNMP is enabled in all the egress routers. (iv) Pathneck ([62]), an open source tool to locate bottleneck links and estimate available bandwidth.

The manipulation of the routing table of the router, monitoring of links and the algorithm to select the egress link were implemented using Perl ([7]) programs.

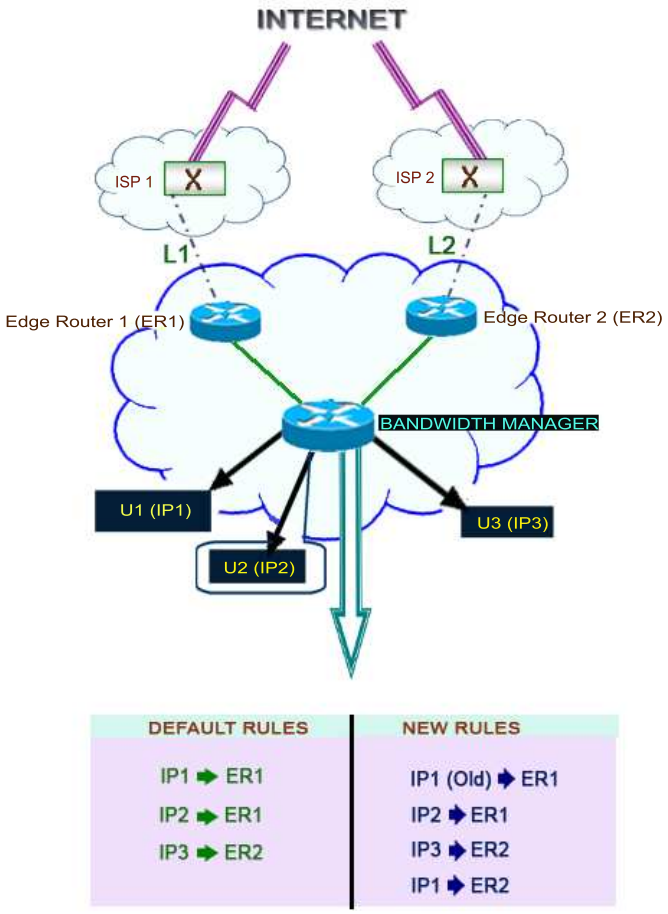


Fig. 4.4 Multihoming Implementation. ER1 and ER2 are edge routers of ISPs, ISP1 and ISP2 respectively.

4.5.1 Measuring Available Bandwidth of the Connecting Path

We SNMP probe each of the edge routers to obtain the available bandwidth of the egress links. The metrics so obtained will be the actual available bandwidth of the *connecting path*, if the first hop connecting the edge router of our network to the provider's edge network is the *connecting path*. Assuming there is only one link that connects our network to the provider's network, this link will always be the first hop of any *connecting path*. Thus our measure of available bandwidth obtained by probing the edge routers is a upper bound

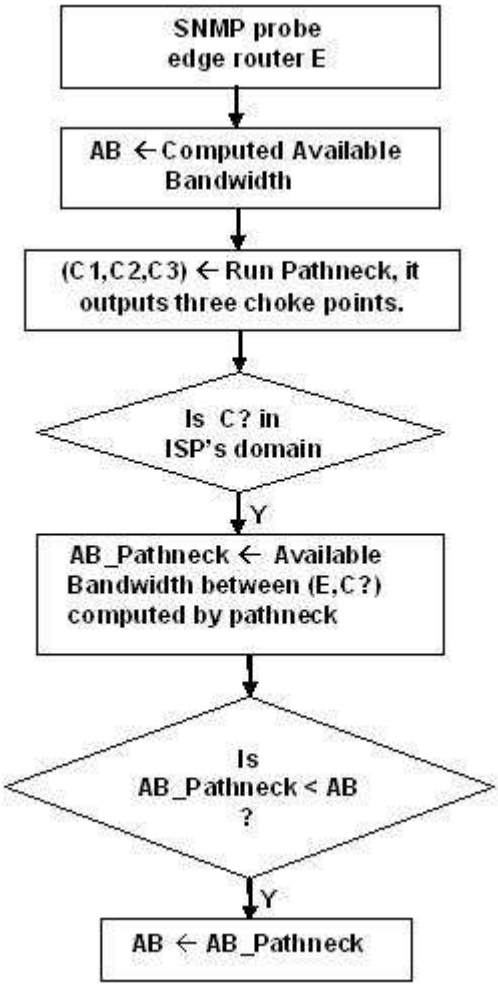


Fig. 4.5 Scheme to monitor connecting path.

on the available bandwidth of the *connecting path*. In order to locate the exact congestion points and refine our measurement of available bandwidth, we use pathneck. This tool when run on Internet paths, outputs three choke points and ranks them based on a computation of *confidence information* of the choke points. We select the first choke point as our congestion point (equivalent to X as mentioned in section 4.3.1), provided the choke point is in the upstream provider's network, otherwise the next choke point is selected. We can intuitively find out the location of a node by observing its domain name. The main drawback of pathneck is that it does not measure available bandwidth of the bottleneck link, but provides a lower or upper bound. The available bandwidth of the bottleneck

link reported by pathneck is compared with our computed upper bound. If the output of pathneck is less, then our assessment of available bandwidth is revised with this value. Figure 4.5 shows the basic operations of our scheme to measure the available bandwidth of the *connecting path*. In our experiments, to get more accurate estimates, the choke points are detected based on 10 probings.

Our load balancing algorithm runs recursively once in every period. In order to ensure that the algorithm get the latest network state in every iteration, the procedure to compute the available bandwidth also loops once every period. The duration of a period is set to 5 minutes. We have tried to keep the durations small in order to handle the chaotic nature of Internet traffic at small time scales. At the same time the period has been kept large enough keeping in mind the re-assignment cost. The other reason behind our choice of the period duration is that typical network operations collect link level statistics every 5 - 15 minutes ([48], [82], [14]). The duration of an interval is kept large enough to allow polling of all the network elements during an interval. On the other hand, the interval should not be too small so as to avoid overloading the polled network elements.

4.5.2 Directing User Traffic to Egress Links

Initially users are assigned egress routes as per the existing static routing policy of the organisation. In this chapter since we focus on small networks, we follow the guidelines of our NAT-based network model, as deliberated in chapter 2. Each user of the end network is assigned a private IP address or an IP address that does not have direct access to the Internet. The routing table of the centralised bandwidth manager is manipulated to direct the traffic of a user to a specific egress route. NAT rules are used to translate the private IP address of a user to the corresponding IP address of its assigned egress link. Let us consider an enterprise network (figure 4.4) with three users and two egress links. Suppose user class U1 with IP address IP1 is assigned egress link 1, user class U2 with IP address

IP2 is also assigned link 1 and user class U3 with IP address IP3 is assigned link 2. In order to direct the user traffic accordingly, the router inserts an *ip rule* for each of these three user classes - *From IP1 lookup T1*, *From IP2 lookup T1* and *From IP3 lookup T2*, where T1 and T2 are the routing tables for egress links L1 and L2 respectively. Let ER1 and ER2 be the IP addresses of egress links L1 and L2 respectively. The following three NAT rules are added to translate the private IP address - *NAT IP1 to ER1*, *NAT IP2 to ER1* and *NAT IP3 to ER2*.

4.5.3 Re-directing User Traffic

After measuring the available bandwidth, the egress route of a user is re-assigned if the utilisation of its assigned egress link is above the link threshold. Suppose utilisation of link L1 has crossed its threshold limit and that of link L2 is below its threshold. We need to switch the user U1 from link L1 to L2. To preserve the ongoing data transfers of the user U1, we first track all its active connections. In Linux, the module "ip_conntrack" maintains a table of all the current connections on the system. The whole list of active connections can be accessed by reading a kernel file (usually /proc/net/ip_conntrack). From this active list, we select those connections that belong to user U1, mark them using NAT rules and then add a *ip rule* to allow the marked connections to continue on link L1. Next we delete the routing table entry and NAT rule for user U1 and add new entries to direct the traffic of U1 to link L2. After these changes, traffic of any new session initiated by the user U1 will follow link L2. The default rules and new rules are shown in figure 4.4.

4.6 Experimental Evaluation

In this section, we explain the experiments conducted using our prototype implementation in a campus network. These experiments demonstrate the practical benefits of multihoming. We show that for a given network, we can empirically determine the user class sizes and threshold limits which will result in overall improvement of Internet download speeds. The

experiments were conducted on different days of the week and at different times of the day to ensure that our model is evaluated against the gamut of network traffic.

4.6.1 Comparison Metric

To evaluate our approach we use the response time of downloads to the local network from Internet servers as a yard stick. Let the latency to access a site x through link y be T_{xy}. Given a site x, the ideal latency T_{xIdeal}, to access the site x can be defined as

$$T_{xIdeal} = \min_{y} T_{xy}$$

where $|y|$ is the total number of external links. If T_{xMulti} is the latency achieved by applying our proposed multihoming approach, then the performance of our route control approach can be measured by comparing T_{xMulti} with T_{xIdeal}. Measuring the ideal latency to access each destination is an infeasible proposal. We, therefore, collect response times assigning users to egress links as per the static routing policy of the organisation. We call such latencies the default response times, $T_{xDefault}$ and the static routing strategy as the default link settings. Next, we collect response times T_{xMulti}, using our load balancing route control technique. Thus our comparison metric is

$$R_{scheme} = \frac{T_{xMulti}}{T_{xDefault}} \tag{4.1}$$

4.6.2 Experiment 1: Performance benefits due to K-Multihoming

A study ([21]) has shown that by carefully choosing the best set of ISPs, subscribers can achieve significant improvement in performance as compared to single homing. In this experiment, we show that similar benefits can be realised if single-homed users are allowed to utilise the unused bandwidth of a second Internet path. Moreover, unlike in [21], where the choice of ISP is crucial, in our approach we use the ISPs that are available at our

disposal. We also find that our scheme, while benefiting a group of users, does not affect the performance of other users. In order to highlight the benefits of K-multihoming, the number of ISPs considered in this experiment was 3. Although this experiment was first performed in the year 2005 we could not test for 3 ISPs due to limitation in the number of ISPs available in our network. The experiments were again repeated in the beginning of 2008.

For this experiment, we considered a client population of about 120. The users were arranged into groups. The addresses of user classes were grouped into single routing table entries using classes interdomain routing (CIDR) address blocks. Let L1, L2 and L3 be the three egress links. In the first execution, all the users were statically assigned to each of the egress links in turn and 24-hour traffic traces were collected. In the second run, we considered two egress links. Our route control scheme was used to influence the routes followed by the users and distribute the traffic among the two egress links. In the third run, load balancing of the user traffic was done considering all the three egress links. The average latencies experienced by the users for all the three executions are shown in figure 4.6. The average RTTs improves by about 10 percent when the users were allowed to use two ISPs and by 15 percent when three ISPs were used. While the perceived download speeds of the user increased there was also an in increase in the average throughput of users. The properties of the trace collected during the experiment are shown in table 4.1.

User Characteristics

The nodes that we have chosen for this experiment were computers located in a lab. The users were mostly students. The computers were not pre-assigned. Users were free to use any of the terminals. Our observation was that on weekdays and during day time the usage was less due to other academic activities of the user. The usage was relatively high on weekends and during night hours. In our attempt to capture the complete user behavior,

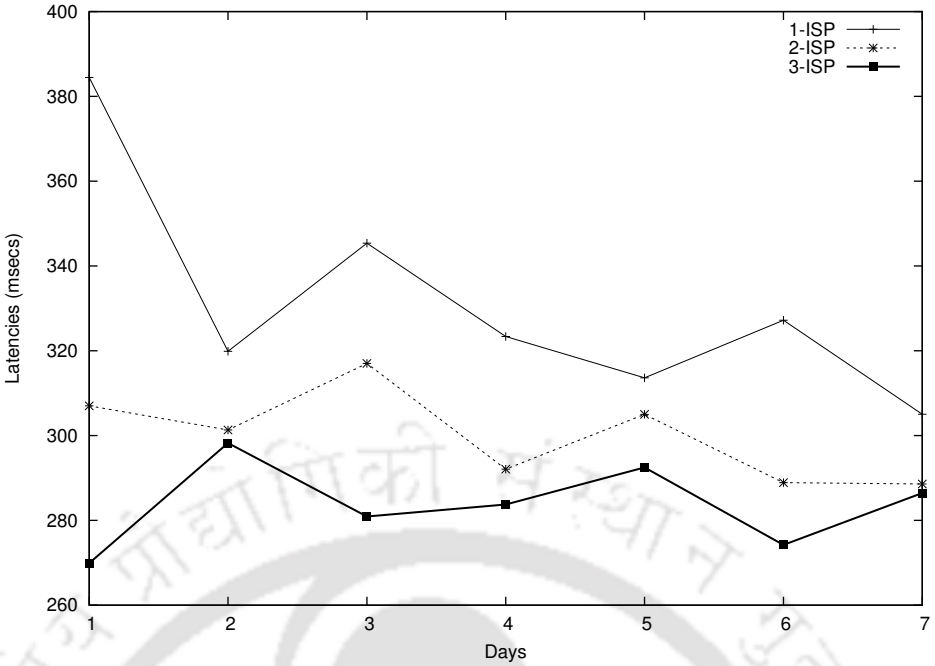


Fig. 4.6 Improvements in latencies due to K-multihoming.

each run of the experiment spanned over all days of the week. In order to determine the user group size, we experimented with different values and selected the largest possible group size. We found that for this experiment a group size of 32 gave excellent performance benefits. There were also other users in the network using these egress links. The combined population of users was about 2000. The routes of other users were left undisturbed.

Table 4.1 Properties of Network Trace collected from first experiment

Experiment Run	**Period**	**Average RTT (msecs)**	**ISPs Used**
Run 1	1 week	331.29	1
Run 2	1 week	299.55	2
Run 3	1 week	283.72	3

Measuring the effectiveness of Multihoming-based Route Control

We quantify the effectiveness of our route control technique by measuring the number of route changes that occur, as a percentage of the total number of periods in the experiment. One would say that our route control scheme is effective, if the number of route changes

is small. However, if the initial static route assignments of the users are skewed, than a single route switch would result in good performance. Our route control scheme basically intends to momentarily steal the unused bandwidth of other links. Therefore, to measure the effectiveness of our scheme, we kept a count of the number of times a user moved to a different under-utilised link and then returned to its default link. Ideally the two values should be equal when the experiments are run for a considerably long period. In figure 4.7 we plot the average percentage of users re-assigned using clustered histograms. The overall user oscillations are about 5 percent when 2 ISPs are used and 10 percent for 3 ISPs. The number of user oscillations seems to be high. The percentage of re-assignments will depend on the total number of users present but more importantly it will depend on the duration of the trace. In this experiments we have considered 24 hour traces. The number of users re-assigned per hour is less than 0.5 percent.

Figure 4.7 does not give the count a user moves out and return to its default link. In the case of a network multihomed to two ISPs, the users can move back and forth between the two links only. Thus if we count the number of times a user moves out and returns to its default link, it should be ideally equal. However, in the case of a network multihomed to three or more ISPs, a user may move out of its default link and then keep on oscillating between the other links. To get a clear picture of the dynamics of the user re-assignments, we plot the oscillations for the first few periods of a user that is multihomed to 3 ISPs in figure 4.8. An upward arrow indicates that the user has been assigned from its default link to a different link at that indicated period. A downward arrow indicates the period at which the user is re-assigned to its default link. Double headed arrows indicate that the user has either moved from the second to the third link or vice-versa. For this particular user shown in figure 4.8, it moved out from its default link in the first period then it oscillated between the second and third link in periods 19, 21, 24 and 43. The user finally returned to its default link in the 45-th period and so on the process continues. The count

of the upward and downward arrows is the same. Further, from the figure we can see that once a user moves to a link it does not immediately move out from the link in the next period but remains assigned to it for at least a few periods. This indicates that our static or default assignments of users to egress links were not skewed. The other point that the figure 4.7 highlight is that due to the dynamic nature of the traffic, re-assignments will be continually required. Thus the load balancing process has to be recursive.

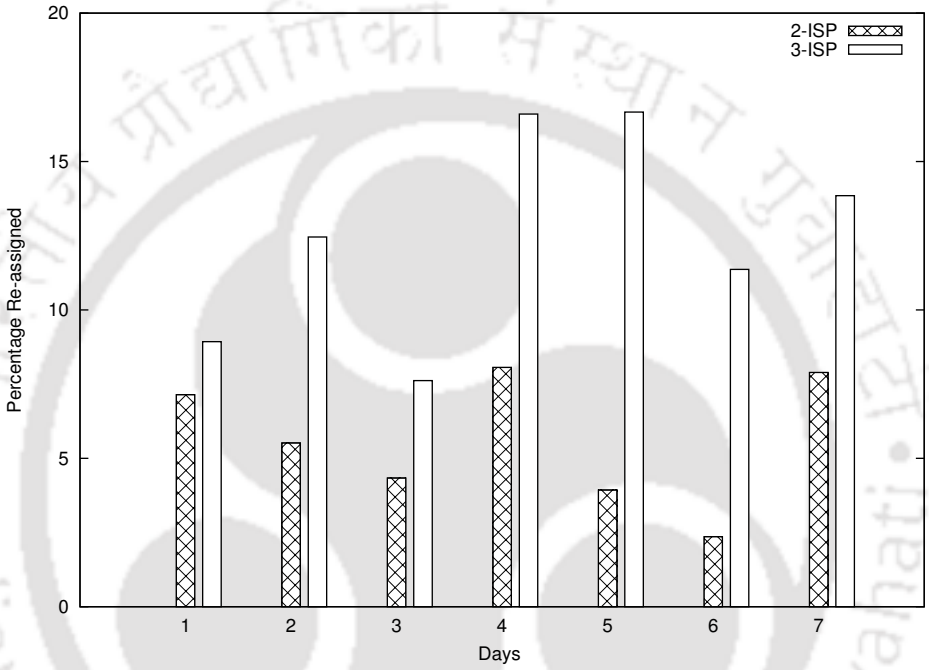


Fig. 4.7 Overheads of K-multihoming, User Re-assignments.

Overheads of Multihoming-based Route Control

The two main side-effects of our multihoming-based route control technique are - (i) overhead of monitoring and (ii) overhead of switching users. In order to actively monitor the links we need to send probe packets. Such probe packets can choke already congested links. Therefore, it is essential for our measurement tool to estimate the bandwidth with minimal number of packets. We have already seen in chapter 3, pathneck is a light-weight monitoring tool and does not send too many probe packets. The major overhead of our route

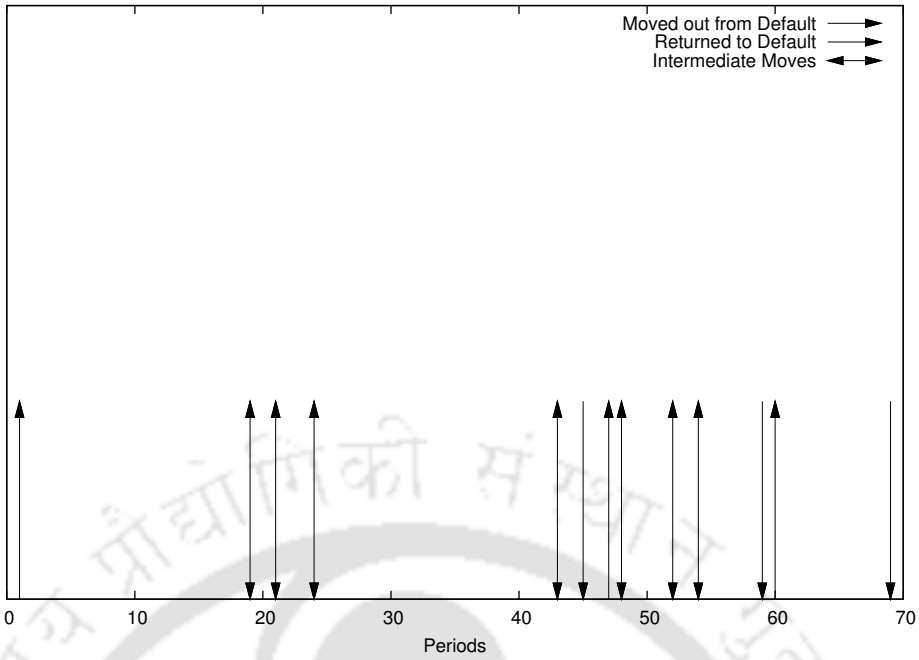


Fig. 4.8 Tracking the link re-assignments of a user multihomed to 3 ISPs.

control technique will be because of making frequent updates to the routing and NAT tables while switching users. In order to preserve active connections, we need to insert additional entries to the routing and NAT tables. These additional entries mainly contribute to the growth of the tables. One way to check the growth of the tables due to these additional entries is to use some filters to delete those entries whose corresponding marked (flagged) connections have completed.

Weakness of live experiments

A major disadvantage when working with real-time traffic is the inability to reproduce the same traffic conditions at a later stage, that is to repeat an experiment with the same input but with a different set of parameters. In order to evaluate our work we need to execute each experiment at least twice - once with static route assignment and a second time with our load balancing route control scheme. To ensure that the traffic conditions in all the executions of an experiment were comparable, the experiments were executed for a sufficiently long period. We also tried to ascertain that the user activity was more or less the

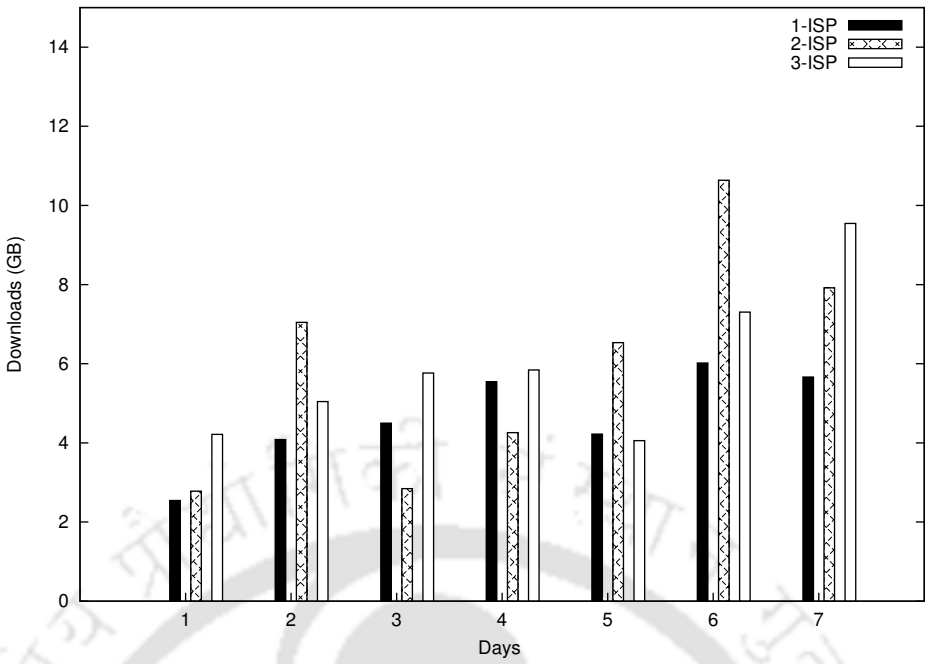


Fig. 4.9 Bytes downloaded using 2 and 3 ISPs.

same during all runs of the experiment. In spite of all our efforts, we cannot guarantee that the traffic conditions will be the same. In figure 4.9, we plot the bytes downloaded day-wise for the three scenarios - 1ISP, 2ISP and 3ISP. As we can see from the plots even when using the same number of ISPs, the bytes download widely varied on different days. To get a more realistic comparison, using the default route assignment we repeated the experiment several times and collected the average response times ($T_{default}$). Next we repeated the experiment with the same setting but using our route control technique for an equal number of times and collected the average response times (T_{xMulti}). We then compared each T_{xMulti} with all the $T_{xDefault}$. Thus our comparison metric given in 4.1 changes to the metric given below. In all our subsequent experiments we use this comparison metric.

$$R_{scheme} = \frac{T_{xMulti}}{\forall(T_{xDefault})} \tag{4.2}$$

4.6.3 Experiment 2: Effects of Large User Classes and Conservative Thresholds

In the first experiment, we have shown the practical benefits of multihoming by using relatively small user class sizes. With this experiment, we show that there is room for improvement even when a very coarse granularity of load balancing is used. The other reason behind our choice of such unreasonably large user class sizes is due to practical compulsions. Organisations generally access the Internet using proxy servers. Network administrators use a number of proxy servers to load balance user traffic among the multiple egress links. In this experiment, we showed that given a set of user classes, we can achieve an over all improvement in performance by carefully choosing the threshold values.

We considered a population size of about 1100 users and two egress links L1 and L2. Although both the links have a capacity of 2 Mbps, Link L1 is a dedicated link, while link L2 is in shared mode with a sustained bandwidth rate of 1 Mbps. Users accessed the Internet using proxy servers. The number of proxy servers used was 4 and the users were distributed among these proxy servers. Keeping in view the link capacities and the number of users present in the proxy servers, static routes were assigned to the proxy servers. We selected two of the proxy servers - Proxy1 and Proxy2, to participate in our route control scheme. The other proxy servers used static route assignment. In table 4.2 we show the size of the proxy servers, their default link assignment and the routing scheme used. The users considered in these experiments belong to the student community. The traces were collected when the network traffic load was high. The duration of a trace was 8 hours (5pm - 1am).

Threshold Settings

In the first execution of our experiment, both the proxy servers were statically assigned to their default link. With this static setting, the experiment was run for five days and network

Table 4.2 User class settings

User Class	Class Size	Default Link Assignment	Routing Scheme Used
Proxy1	300	L1	Load Balancing
Proxy2	200	L2	Load Balancing
Proxy3	200	L2	Static
Proxy4	400	L1	Static

traces were collected. In table 4.3, we show day-wise the latencies experienced and other statistics of the traces collected. Next, using our load balancing algorithm, we repeated the experiment for another five days and collected traces. The threshold values were set based on the past performance of the links. Link L1 being a superior link its threshold value was set in the range between eighty to ninety percent. While for link L2 the threshold value was set in the range of forty to fifty percent. The properties of the trace collected using our load balancing algorithm are shown in table 4.4. The users that we considered in these experiments were mainly students. During vacations or examinations, the download rate of the students was greatly reduced due to obvious reasons. Moreover, there were also days when Internet connectivity was disturbed due to link failures. Taking into account such factors, the experiments were conducted only on those dates when Internet browsing activity was normal and students were present in the campus attending their regular classes.

Unraveling the performance degradation

As can be seen from table 4.3 and table 4.4, the application of our load balancing has resulted in a degradation on the performance of Internet downloads. To get a better view, we plot a comparison of the RTTs of the proxy servers, using our comparison metric given in equation 4.2. The comparison of the latencies is shown in blocks in figure 4.10. A block shows the comparison of each value of T_{xMulti} with all values of $T_{xDefault}$. Thus there are five comparison blocks for the five different traces and in each block there are five histograms. For example, in the first block the RTT of 27.7.2005 (with load balancing) is compared with the RTTs of 30.7.2005, 6.8.2005, 19.8.2008, 21.8.2005 and 24.8.2005 (all

Table 4.3 Properties of trace (without load balancing)

Date	**User Class**	**Average RTT** $T_{xDefault}$	**TCP Connections**	**Bytes Download**
30.7.2005	Proxy1	1095.2	351181	1313460224
	Proxy2	510.33	193657	332823915
	Proxy3	460	Not Measured	Not Measured
	Proxy4	1200.3	Not Measured	Not Measured
6.8.2005	Proxy1	934.42	352380	1037684736
	Proxy2	670.46	191053	312461312
	Proxy3	472.36	Not Measured	Not Measured
	Proxy4	1304.23	Not Measured	Not Measured
19.08.2005	Proxy1	1095.34	477457	1550823424
	Proxy2	530.58	261156	405204992
	Proxy3	467.38	Not Measured	Not Measured
	Proxy4	1192.2	Not Measured	Not Measured
21.08.2005	Proxy1	1020.6	281390	1074604774
	Proxy2	556.16	149116	227937138
	Proxy3	793.47	Not Measured	Not Measured
	Proxy4	1099.36	Not Measured	Not Measured
24.8.2005	Proxy1	924.04	326251	1029411499
	Proxy2	594.41	179947	272498481
	Proxy3	444.52	Not Measured	Not Measured
	Proxy4	1004.33	Not Measured	Not Measured

without load balancing). If the value of the comparison ratio is less than 1, it indicates an improvement in performance and a value greater than 1 indicates degradation. We find that the performance as a whole has degraded. The other interesting observation is that the performance of the smaller proxy server (Proxy2) has degraded substantially while the degradation of the larger proxy server (Proxy1) is relatively less. The average performance of Proxy2 has degraded by more than 15 percent while that of Proxy1 has degraded by about 2 percent.

The performance of our algorithm depends on three factors - (i) precision of the link monitoring algorithm (ii) size of user classes and (iii) link threshold values. In theory, if all of these three parameters are exact, our route control scheme as compared to static routing should either show an increase in performance or the performance should remain unchanged

Table 4.4 Expt 2: Properties of traces (with load balancing)

Date	**User Class**	**Average RTT** T_{xMulti}	**TCP Connections**	**Bytes Download**
27.7.2005	Proxy1	1211.5	194004	877926813
	Proxy2	699.04	110124	233868151
	Proxy3	647.86	Not Measured	Not Measured
	Proxy4	1290.2	Not Measured	Not Measured
29.7.2005	Proxy1	1063.04	260932	961103885
	Proxy2	690.04	137313	285159629
	Proxy3	731.68	Not Measured	Not Measured
	Proxy4	1190.4	Not Measured	Not Measured
31.7.2005	Proxy1	912.68	129609	509804544
	Proxy2	596.6	77780	141100013
	Proxy3	858.98	Not Measured	Not Measured
	Proxy4	1090.2	Not Measured	Not Measured
5.08.2005	Proxy1	1134.08	327598	962826240
	Proxy2	699.7	151094	311447552
	Proxy3	824.95	Not Measured	Not Measured
	Proxy4	1290.6	Not Measured	Not Measured
11.08.2005	Proxy1	898.11	224890	667672576
	Proxy2	745.95	112858	215773184
	Proxy3	867.81	Not Measured	Not Measured
	Proxy4	996.2.2	Not Measured	Not Measured

but should not degrade. Accurately estimating the available bandwidth of the bottleneck link is an active area of research. As we have seen in chapter 3, as of date researchers have not been able to design a tool that works accurately on most Internet paths. A more detailed discussion on how to load balance traffic in the presence of imprecise network information is given in chapter 5. Assuming that the errors in link measurements are within acceptable limits, the degradation in performance will be either because of faulty user class sizes or threshold values. Further keeping in view our practical limitations, we assumed that the size of the user classes cannot be altered. In order to measure the effects of threshold settings on the performance, we plot the number of re-assignments for both the proxy servers (figure 4.11). The overall re-assignment for Proxy1 was 3 percent while that of Proxy2 was 9 percent, a relatively high figure. At the same time, the number of re-assignments for Proxy2 was not that high to suggest that the user had been constantly

oscillating between the two links. The degradation in performance of Proxy2 in spite of a higher number of re-assignments suggests that our estimate of the threshold limit for link L2 had been conservative. Due to the low threshold value of link L2, Proxy2 frequently switched from its default link L2 to link L1, a relatively more congested link. The other possibility was that the threshold limit of link L1 had been over-estimated, due to which it tried to accommodate an user which in reality it could not.

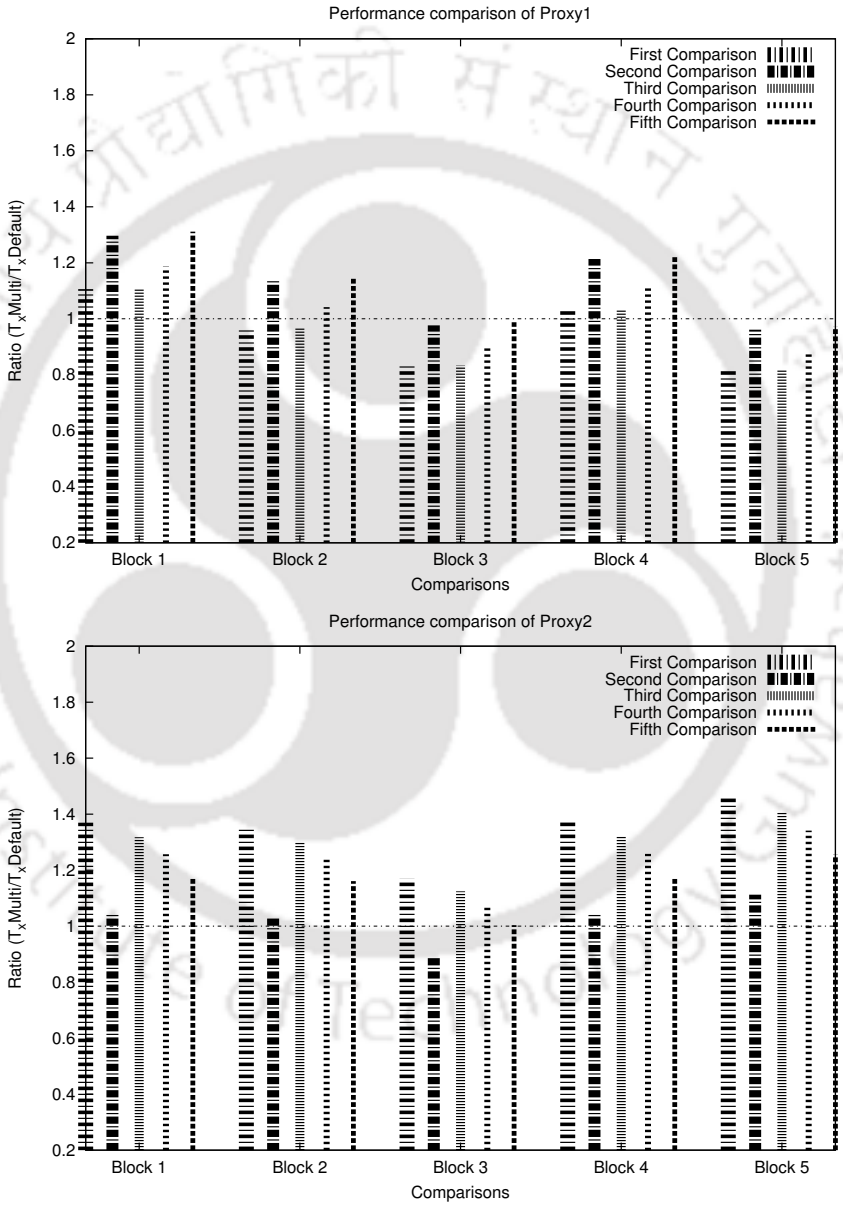


Fig. 4.10 Expt. 2: Comparison of link utilisations.

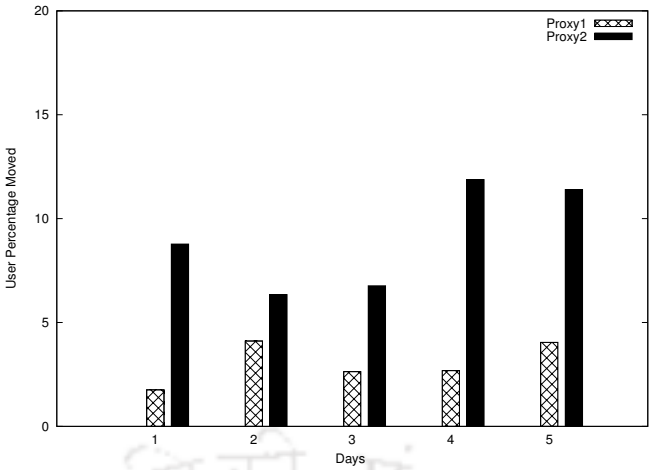


Fig. 4.11 Expt. 2: Comparison of user switches.

4.6.4 Experiment 3: Extracting Performance Improvements by Fine-tuning Threshold Values

We repeated the previous experiment, by raising the threshold limit of the second link L2 on an average by about 5 percent, while keeping all other parameters constant. Using our load balancing scheme, we collected traces for another 5 days. The raise in the threshold limit is marginal, since we do not want to over estimate. The properties of the trace collected are shown in table 4.5. In figure 4.12, we compare the latencies of the proxy servers with their corresponding latencies collected earlier using static routing (table 4.3). With the new threshold settings, we found that although the performance as compared to static routing was still inferior, the performance of Proxy1 had improved in comparison to the results of our previous experiment. The performance of Proxy2 had also marginally improved. This improvement in performance motivated us to further manipulate the threshold values.

On examining the user re-assignments (figure 4.13), we found that by increasing the threshold limit for link L2, the number of re-assignments for Proxy1 increased (5.48 percent) while that of Proxy2 got reduced (7.35 percent). After we have increased the threshold limit of link L2, the frequency of utilisation of link L2 exceeding its threshold value had been

reduced. As a result Proxy2 got re-assigned less frequently. At the same time since link L2 remained below its threshold value during more number of periods, it could accommodate Proxy1 more often. This is why the number of re-assignments for Proxy1 had increased. Due to the higher number of re-assignments, the performance of Proxy1 had improved which indicated that Proxy1 was getting re-assigned to a more superior link. We further increased the threshold limit of link L2 and repeated the experiment.

Table 4.5 Expt 3: Properties of traces (with load balancing)

Date	**User Class**	**Average RTT** T_{xMulti}	**TCP Connections**	**Bytes Download**
12.8.2005	Proxy1	936.5	377704	971284480
	Proxy2	629.29	167721	329060352
	Proxy3	664.95	Not Measured	Not Measured
	Proxy4	921.4	Not Measured	Not Measured
16.8.2005	Proxy1	1091.65	282641	830962595
	Proxy2	779.8	164479	301478302
	Proxy3	805.7	Not Measured	Not Measured
	Proxy4	1212.3	Not Measured	Not Measured
17.8.2005	Proxy1	1273.93	346136	995954123
	Proxy2	585.9	185284	351081553
	Proxy3	539.45	Not Measured	Not Measured
	Proxy4	1083.23	Not Measured	Not Measured
18.8.2005	Proxy1	910	218562	597402244
	Proxy2	627	118936	197404526
	Proxy3	741.05	Not Measured	Not Measured
	Proxy4	1103.43	Not Measured	Not Measured
27.8.2005	Proxy1	953.36	593190	1647411200
	Proxy2	776.62	482163	739888383
	Proxy3	550.65	Not Measured	Not Measured
	Proxy4	1215.6	Not Measured	Not Measured

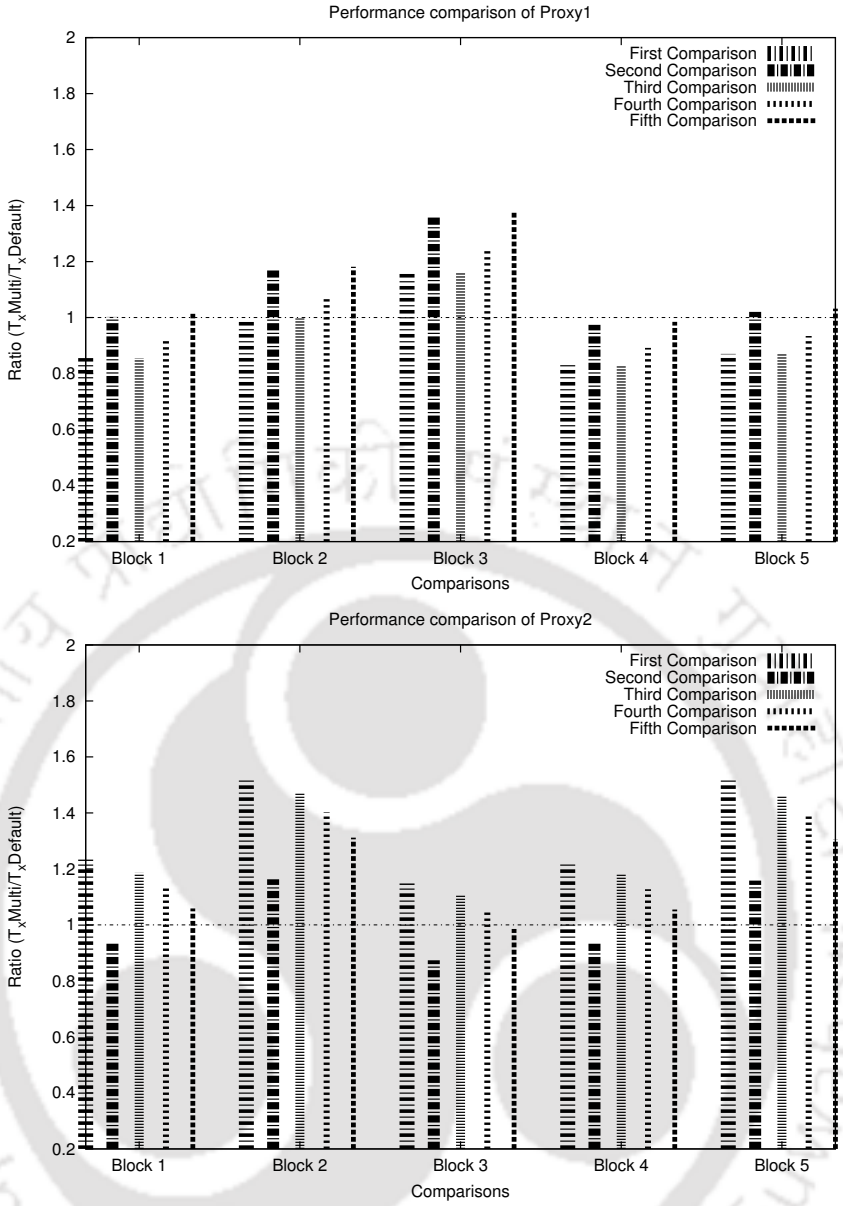


Fig. 4.12 Expt. 3: Comparison of link utilisations.

4.6.5 Experiment 4: Achieving Overall Performance Improvement with Large User Classes

After experimenting with various threshold limits we finally arrived at a value that gave an overall improvement in performance. We found that increasing the threshold limit of link L1 did not prove beneficial. The threshold limit of link L2 was set in the range of 55 to 60 percent. The properties of the traces collected are given in table 4.6. In figure 4.14,

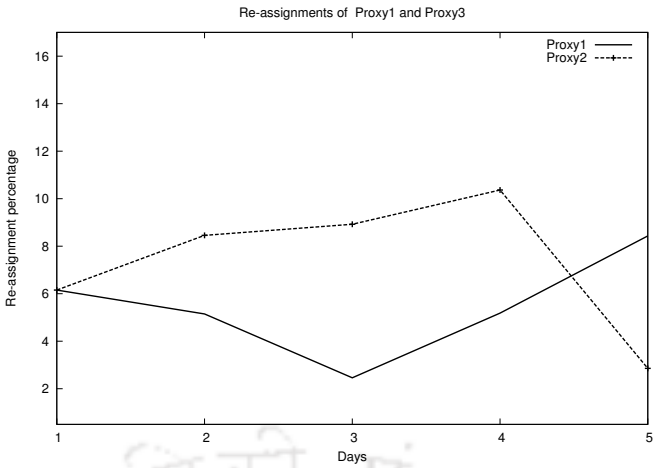


Fig. 4.13 Expt. 3: Comparison of user switches.

we compared the latencies achieved using the comparison metric given in equation 4.2. Proxy1 showed considerable improvement in performance while there was still a marginal decrease in performance for proxy2. The percentage of re-assignments for the two proxy servers is shown in figure 4.15. Due to our increase in the threshold limit of link L2, Proxy1 now moved over more frequently to link L2, while the re-assignments for Proxy2 had been greatly reduced. The number of re-assignments for Proxy1 was about 8.8 percent while that of Proxy2 was about 4 percent. On further examination, we found that due to the large size of Proxy1 whenever it moved to link L2, the link crossed its threshold limit within a few periods and thus Proxy1 needed to move back. This is why the number of re-assignments was high for Proxy1. In spite of such momentary re-assignments for Proxy1, we found that its performance improved by about 10 percent. The performance of Proxy2 degraded (by about 3 percent) because link L1 was already a congested link, therefore moving a user into link L1 did not prove to be beneficial. On hind-sight, we should have statically assigned routes for Proxy2 while allowing Proxy1 to oscillate between the two links.

Table 4.6 Expt 4: Properties of traces (with load balancing)

Date	User Class	Average RTT T_{xMulti}	TCP Connections	Bytes Download
31.8.2005	Proxy1	996.52	258415	799008512
	Proxy2	647.65	163865	251470654
	Proxy3	753.15	Not Measured	Not Measured
	Proxy4	1202.2	Not Measured	Not Measured
1.9.2005	Proxy1	835.17	274874	752003310
	Proxy2	562.21	153905	261831256
	Proxy3	770.76	Not Measured	Not Measured
	Proxy4	830.2	Not Measured	Not Measured
2.9.2005	Proxy1	792.04	253779	695348291
	Proxy2	550.32	142914	235362434
	Proxy3	434.5	Not Measured	Not Measured
	Proxy4	800	Not Measured	Not Measured
3.9.2005	Proxy1	850.5	344822	1259174358
	Proxy2	595.93	120157	216517305
	Proxy3	495.87	Not Measured	Not Measured
	Proxy4	807.6	Not Measured	Not Measured
5.9.2005	Proxy1	830.41	268701	834585088
	Proxy2	597.79	120259	203628544
	Proxy3	520.9	Not Measured	Not Measured
	Proxy4	932.2	Not Measured	Not Measured

4.7 Conclusion

The goal of this chapter was to show the benefits of traffic load balancing in a real network. The route control strategy relies on a simple threshold-based mechanism. The threshold value of a link is estimated on the basis of the past performance of the link. The most current samples have been used to estimate the link thresholds. When the utilisation of a link exceeds its threshold value, some users from the link are re-assigned to the *least* utilised link. The key advantage of our scheme is that we try to optimise resources that are at the disposal of the local network only and we do not rely on any third party assistance. The other advantage is that when the traffic of a user is re-assigned, we ensure that its active connections are not disturbed. This makes our scheme transparent to the end users. Our focus in this work was on multihomed LANs. We implemented a prototype of our route

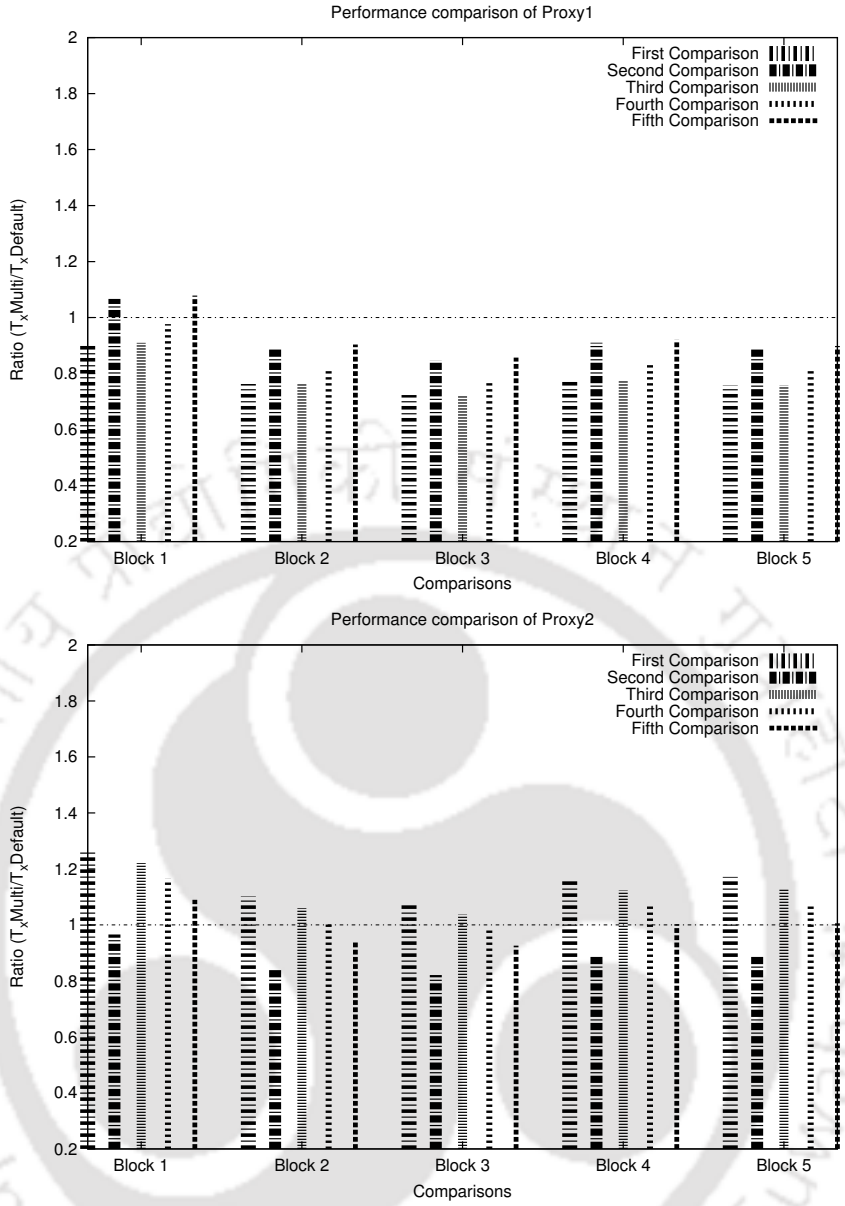


Fig. 4.14 Expt. 4: Comparison of link utilisations.

control technique on a Linux-based server and deployed it on our institute campus. In the first experiment we highlighted the benefits of traffic load balancing in a multihomed network using three ISPs. Our experiments showed that round-trip time improved by 10 percent when two ISPs were used as compared to using a single ISP. The performance benefit was about 15 percent when three ISPs were used. The number of re-assignments required to load balance the traffic ranged from about less than 0.5 percent.

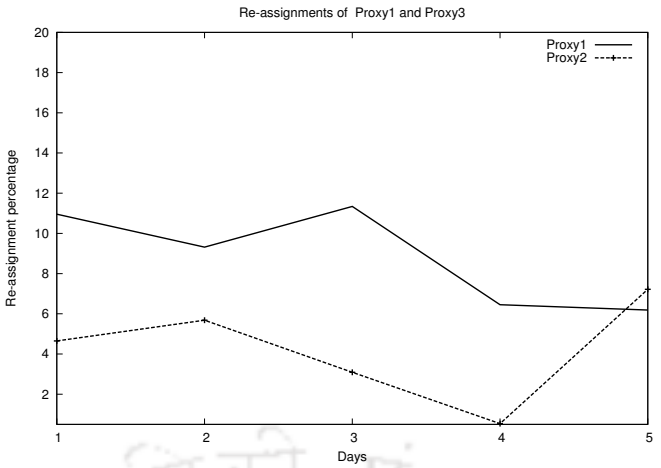


Fig. 4.15 Expt. 4: Comparison of user switches.

In the later experiments we demonstrate how performance improvements can be achieved even with a very coarse granularity of load balancing. Organisations generally access the Internet using proxy servers. We tried to load balance the traffic by re-assigning the egress route of the proxy servers. The size of the user classes in these experiments ranged from 18 to 36 percent of the total user population, whereas in the first experiment the user class size was about 2 percent. In general, a multihomed network balances its traffic by manually assigning its users to the different egress links. We call such a user assignment the *default* approach and our load balancing mechanism was compared to this *default* approach. The overall improvement in performance realised was about 7 percent. The total number of re-assignments required per hour was less than 2 percent. The other point that we prove with these experiments is that the benefits of our multihoming route control can be realised without disturbing the existing setup of a network. We also showed that the feedback from empirical results can be used to further fine-tune the threshold settings.

In this chapter we have shown the practical benefits of load balancing by considering a small network. However, as the size of the network grows, the volume of Internet traffic as well as effects of the intradomain traffic will increase immensely. Determining the link

thresholds will become more difficult. The threshold-based route control mechanism therefore will not be effective in such *large* networks. We try to address these issues as well as provide an in depth analysis of load balancing traffic in the next chapter.

Chapter 5

Load Balancing in Hop-by-Hop Networks

Interdomain traffic engineering implements rules for routing data according to a view of current resource availability and the expected traffic volume. The basic aim is to balance the traffic load over various links so that none of the links are over-utilised or under-utilised. Traffic engineering may be a manual process, where a network administrator monitors the state of the network and routes traffic or provisions additional resources. Alternatively, the process may be automated, wherein the system reacts depending on the information provided. In the previous chapter we provided a comparison between a manual process of traffic engineering and an automated one. Our studies reveal that in a multihomed environment, traffic along one path may experience congestion even when other paths are under-utilised. Significant improvement in download speeds can be achieved by re-balancing the traffic load. However, to keep the traffic load balanced, frequent updates to the routing table is required and it is not feasible to do so manually. Yong Liu and A.L.N. Reddy ([82]) studied the characteristics of alternate paths in multihomed stub ASes. In their study they concluded that in order to improve performance, route control should be dynamic. Moreover, as the size of the network increases and BGP is used for interdomain routing,

changing routes manually will require a high level of expertise. In the preceding chapter, we demonstrated the benefits of automating the process of load balancing using a simple threshold-based approach. Our approach, although simple relies on the correct definition of certain key parameters. These parameters cannot always be accurately estimated as an end network has limited knowledge about user demands, available networks resources and routing policies of other competing networks. For a small network these parameters can be empirically determined, but as the size of the network grows, limitations configuring these metrics will become more and more difficult.

A major challenge in interdomain traffic engineering is due to the level of uncertainty faced by networks. With many potentially competing networks each trying to optimise its own resources, influencing the traffic entering a network can be very difficult. Moreover, existing network routing protocols have not been primarily designed from an interdomain traffic engineering perspective. For wider acceptance, the need to propose solutions within the framework of existing protocols and standards makes the problem even more challenging. In medium ISPs and corporate networks, the network topology is simple and their egress links are usually over-provisioned. In such networks, the costly resource that needs to be optimised is their interdomain connectivity. Nevertheless, the intradomain paths too can become a source of bottleneck. Typically, the network will spread over a large geographical area with leased lines used to inter-connect the LANs of the organisation at the different locations. Traffic within the network will be routed using a hop-by-hop routing paradigm like OSPF. In such large networks, the intradomain traffic path from a node to an egress link can be several hops. The internal network traffic, therefore, can have a substantial effect in the overall latency ([90]). Our earlier assumption (for LANs) that internal bandwidth is large and delay and packet loss rate within the network are negligible as compared to those on egress links, will no longer hold.

In this chapter, we intend to tackle the unpredictability involved in interdomain traffic by addressing the limitations of the threshold-based approach. Additionally, we also incorporate the issues of intradomain traffic into our problem formulation. A detailed description of our problem and the problem formulation is given in sections 5.1 and 5.2 respectively. We attempt to solve the problem in two steps. In the first phase, in our endeavour to find an exact solution, we make unrealistic assumptions about the input traffic and bandwidth. We assume that the input traffic is known and bandwidth measurements are accurate. Even with such strong and unrealistic assumptions, we show that the problem is NP-complete. This part of our study is called the *offline analysis*. The offline models are described in section 5.3. In the second phase of our study, we relax all these assumptions and assume a dynamic environment where input traffic is unknown and network measurements are inaccurate. This part of our analysis is called the *online analysis*. Leveraging upon our solutions developed for the offline case, we propose a number of online heuristics. The online models are presented in section 5.4. We study the performance of our proposals using both synthetic as well as real traffic traces. During evaluation of our techniques under different traffic scenarios, we discover certain limitations to our approach. The heuristics are further tailored to address these short-comings. In section 5.5, the evaluation of our proposed route control techniques and results are presented.

5.1 Problem Description

One of the first tasks in interdomain traffic engineering is to locate the bottleneck link and measuring the available bandwidth of the bottleneck path. In the previous chapter we have identified the bottleneck path as the network path that connects the end network to its provider and this path was referred to as the so called *connecting path*. A survey of the open source tools to measure available bandwidth was done in 3 and a mechanism was proposed to measure the available bandwidth of the *connecting path* using currently available techniques. As we found that none of the existing tools can give an exact measure of the available

bandwidth, our mechanism included a check to ensure that the measurements do not deviate beyond a certain upper limit. Nevertheless, we cannot assume that our measurements are accurate. In this chapter, we assume measurement of available bandwidth of the links is approximate. The formulation of our problem is strongly motivated by the limitations of currently available tools to measure network metrics.

5.1.1 Challenges in the Threshold-Based Approach

Let us visualise a situation where we directly deploy the threshold-based route control technique described in chapter 4 in a large end network. The challenges that will confront the approach will be mainly on the following three fronts. A major part of our problem formulation is devoted to addressing these issues.

- **Measure of Link Utilisation :** The switching of users depend on the current utilisation of the egress links. In case of an incorrect estimate of utilisation, users may get moved though actually they should not or vice-versa. However, as we have seen currently available tools cannot guarantee that measurement of available bandwidth is accurate at all time instants.

- **Dynamic Thresholds :** The efficiency of the threshold-based approach to a large extent depends on the precise estimate of link thresholds. Although the capacity of a link is fixed, its instantaneous available bandwidth will depend on cross traffic, intensity of usage and link quality. The threshold value is therefore expected to vary over a wide range and correctly predicting these values for all time instants will be very difficult.

- **Dynamic User Classes :** When a user class is moved the amount of traffic that is switched directly depends on the size of the user class. The traffic of a user class in turn will depend on the intensity of usage of the members of the class. In the threshold-based approach the user classes are pre-defined, therefore, the traffic re-

ceived by the class will vary at different time instants. Ideally we would like to dynamically define the user classes so that optimum amount of traffic is moved.

5.1.2 Network Model and Traffic

The network that we consider is a full-fledged multihomed, stub autonomous system geographically spread over a wide area. We consider that the network has a block of IP addresses independent of its ISP. Networks that have independent IP address block use BGP to advertise the IP addresses among its different ISPs. One possible IP address advertisement plan is to announce the address block on all the egress links. Such a scheme will ensure that if one link fails the hosts will still be reachable through the other links. However, with such an assignment plan, the stub network will have no control on the incoming traffic. The routes of the ingress traffic will be decided by the BGP relationships between ASes. Selective announcement is a popular technique used in BGP to control incoming traffic. The original IP address block is divided into sub-prefixes and each sub-prefix is advertised on a different link.

The most practical way of regulating the incoming traffic would be to use NAT-based solutions. In NAT-based solutions the nodes of the network are assigned private IP addresses and depending on which egress route is chosen for a node, its IP address is translated to the IP address of the egress link. However, regulating the incoming traffic using NAT-based solutions may fail to scale for large networks. In such a scenario, we propose to use a variant of the selective advertisement. Initially the IP address sub-prefixes are announced on the various routes similar to the selective advertisement. Each sub-prefix is assumed to be a user class. As the route of a user is changed from one egress link to the other in order to load balance the traffic, we withdraw its corresponding IP prefix from the previous link and announce it on the new link. We call this network model the BGP-based model. This network model was discussed in detail in chapter 2 (section 2.5.2). We assume that given

this network model, traffic of a user send will enter the network through only one link, the link on which its prefix is announced.

Using the terminology given in the literature ([17]), an AS can have two types of links - backbone and edge. Backbone links are used to interconnect the intradomain routers of a domain. Edge links can be of two types - access and egress. Access links connect to downstream users or customers of the ISP. Egress links connect directly to peers, transit providers or Internet Exchange Points. The various possible links of an AS are shown in figure 5.1. The possible traffic flows are:

i. Internal Traffic : Traffic that travels from one access link to another.

ii. Transit Traffic : Traffic that travels from an egress link to another egress link.

iii. Outbound Traffic : Traffic that flows from an access link to an egress link.

iv. Inbound Traffic : Traffic that travels from an egress link to an access link.

An outbound prefix flow is defined as a set of IP packets whose source address matches one of the ISP's customer IP prefix and the destination IP prefix does not belong to the AS. Similarly an inbound prefix flow is defined as a set of IP packets whose destination matches one of the ISP's customer IP prefix and the source address is an IP address that does not belong to the AS. In this work, since we assume that the main congestion is on the incoming traffic, we only consider the inbound prefix flows in our measurements and performance computations. We make minimal assumptions about the incoming traffic. The basic granularity of load balancing is assumed to be a user (IP address). Typically, a domain will have a large address block and dynamically changing the egress routes of all the users will be cumbersome and unmanageable. To keep the routing tables simple and manageable, we aggregate the users into a user class. In this work each advertised sub-prefix is assumed to be a user class. Users that have similar QoS requirements are

allotted IP addresses from the same prefix.

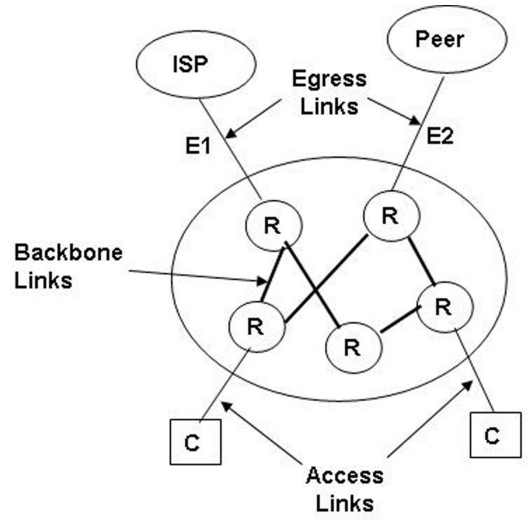


Fig. 5.1 Links of an Autonomous System.

5.1.3 Assumptions

From the above discussions it is evident that we can neither make any assumption about the traffic entering the network nor assume accurate knowledge of the available bandwidth of interdomain paths across which we want to balance the traffic. Further we presume that we cannot depend on any congestion indication signal from the network to initiate our route control algorithm. The only assumption made is that the traffic entering the network can be split at a certain level of granularity. The other assumption is that the overall latency will improve if the traffic is properly load balanced. We have already shown in the previous chapter that load balancing of traffic indeed results in improvement of latencies. In this chapter too, we show the same using synthetic data. Other than this we do not make any assumptions about the characteristics of traffic.

5.1.4 Egress Route Selection

In chapter 2, we have seen the steps involved for a BGP speaking router to select an egress route. If the BGP decision process results in multiple equally good egress routes each router directs traffic to its closest egress route in terms of the IGP distance. Although this BGP policy of *early exit* or *hot-potato routing* guarantees a router's choice of egress point to be consistent with other routers along the forwarding path, it does not consider the egress points from a traffic engineering perspective. Moreover, *hot-potato routing* reacts immediately to small changes in IGP distance which may cause convergence delays due to the frequent disruptions. Selecting different egress routes can have diverse effects on the network resource utilisation. Researchers have proposed a number of models on how to achieve traffic engineering objectives using egress route selection. The critical issue that needs to be addressed is how to select an egress router so that network wide performance objectives are met. In the simplest case, egress points are statically ranked from the perspective of each ingress router. Data packets follow the shortest IGP path from the ingress to the egress router. While such a fixed ranking of egress routers is robust to internal network changes, it cannot adapt to changes in the topology and traffic. In order to incorporate internal topology changes in the selection process, R. Teixeira et. al. ([93]) proposed a tunable egress selection technique. Each ingress router for a given destination prefix computes the egress point based on the IGP distance as well as a static ranking of the egress routers. The packets are forwarded over an IP tunnel that follows the shortest path through the network to that egress point. T.C. Bressoud and Rajeev Rastogi ([106]) proposed heuristics that aims to optimise the cost of carrying traffic over a provider's network while also load balancing the traffic based on the capacity of the egress links. Their aim is to determine the optimal set of border routers for advertisement of network prefixes that will fulfill the two objectives. Kin-Hon Ho et. al. ([77]) explores the possibility of using egress route selection techniques to satisfy end-to-end QoS requirements. The two performance objectives addressed by them are satisfying the customer bandwidth require-

ment as well as minimising the total bandwidth consumption. They propose a number of greedy heuristics to solve the problem.

As is evident from our discussion so far, a stub network has no control or knowledge about its upstream providers and peer networks. The only control it has is in the choice of egress route for outbound traffic from a set of possible paths at a certain level of granularity (in our case an IP address). In this context, our work will broadly fall in the realm of egress route selection. However, our objectives and approach are different from any of the previous works. The focus so far has been on improving the performance of outgoing traffic - bandwidth, quality of service etc. Our egress route assignments are driven by incoming traffic characterisation. In our approach, when the end network notices that a particular egress link is congested, it tries to re-route some of its incoming traffic by regulating the outbound traffic. In this dissertation, our focus being primarily on access networks users will basically download information from the Internet. Requests for downloads will mainly originate from within the network. The traffic pattern for an access service provider will, therefore, typically consists of small requests and large responses. As discussed in chapter 2, the network model proposed in this work ensures that outgoing and incoming traffic follow the same route. Thus if we intelligently schedule the outgoing requests, then the incoming traffic on the different egress links can be re-distributed.

5.2 Problem Formulation

Consider a network with N egress links and let S be the size of the user population. Time is divided into discrete intervals, indexed by t={1,2,...}, a period t denotes the time interval (t-1,t). We now define the notations required for formulating the problem - sets, variables and constants:

Sets:

$E = \{e_1, e_2, ..e_N\}$ denotes the set of egress links.

$C = \{C_1, C_2, ..C_N\}$ denotes the capacity of the egress links.

$I = \{i_1, i_2, ..i_S\}$ denotes the set of users.

$T \in \{t_1, t_2, ..\}$ denotes the set of time intervals.

Variables and Constants:

$A_e(t)$: Available bandwidth (Mbps) of link e at time t.

$U_e(t)$: Utilisation of link (Mbps) e at time t. $(0 \leq U_e(t) \leq C_e)$

$b_i(t)$: Incoming traffic (Mbps) of user i at time t.

$$x_{ie}(t) = \begin{cases} 1 & \text{If user } i \text{ is assigned link } e \text{ at time t-1,} \\ 0 & \text{otherwise.} \end{cases}$$

At any given time instant a user can be assigned to only one egress link, therefore, $\sum_{e=1}^{N} x_{ie}(t) = 1, \ \forall i \in I$. A user i is assumed to be assigned to an egress link even if the instantaneous incoming traffic of the user is nil, i.e. $b_i(t) = 0$.

The objectives that we attempt to achieve here are:

i. *Keep link utilisation in tune with available bandwidth :* One way of optimising Internet performance would be to load balance the traffic based on the capacity of the border router links ([106]). However, the capacity of a link does not reflect the instantaneous traffic carrying capability of the link. We, therefore, state that the traffic is properly load balanced if the utilisation of all the egress links with respect to their available bandwidth, are equal. The corresponding link utilisations are considered as their ideal value. A formal definition of the ideal utilisation of an egress link e is given below, where K for any given period is a constant.

$$\text{Ideal utilisation of link } e \text{ at time t} = IU_e(t) = K \ . \ A_e(t) \tag{5.1}$$

$$where\ K = \frac{\sum_{l=1}^{N} U_l(t)}{\sum_{l=1}^{N} A_l(t)}$$

In practice, we may not be able to accurately estimate the ideal utilisation of a link since our measure of available bandwidth is an approximation. However, the deviation in our estimate of available bandwidth is expected to be between reasonable limits. Thus, if we can ensure that actual utilisation of links are equal to their ideal value, it will result in a fair utilisation of resources even if the measure of available bandwidth is not exact. The advantage of formulating the problem in this form is that we need not explicitly define threshold values for links. Nevertheless depending on the available bandwidth, each link will have a limit on the maximum traffic it can carry (i.e. link threshold). Since our model re-distributes traffic every period based on the available bandwidth, it will ensure that utilisation of all the links are below their threshold values. If at all thresholds are encountered it will be for all the egress links. Our first objective, therefore, is to minimize the difference between ideal utilisation and actual utilisation of a link.

$$f1: \ min|IU_e(t) - U_e(t)| \ \forall e \in E$$

ii. *Minimize user re-assignments :* Given that we no longer wait for a link to reach its threshold value, but re-assign users as soon as its utilisation deviates from the ideal value, the number of user switches is likely to be high. Our next goal is to keep the user re-assignments as small as possible. That is we wish to minimize

$$f2: \ min \sum_{e=1}^{N} \sum_{i=1}^{S} |x_{ie}(t) - x_{ie}(t-1)| \tag{5.2}$$

For each user re-assignment, the output of the above equation will increase by a value of 2.

iii. *Select the minimum cost intra-domain link :* The third objective is to handle the issues of latencies encountered within the network. Let $E' \subseteq E$ denote the set of equally good egress routes to which an user i can be moved. If our algorithm selects $e \in E'$ as the egress route to which the user will be moved and d(i,e) denotes the intradomain cost from user i to egress link e, then the following equation must hold

$$f3: \ d(i,e) \leq d(i,e'), \ \forall e' \in E' \tag{5.3}$$

5.3 Offline Models

One of the main impediments in finding a solution to our problem is that the input traffic takes unknown values at the time of making decisions. In order to exhibit the complexity of the problem we make some simplifying assumptions about the traffic, though some of these assumptions may sound unrealistic. For each such assumption we examine if a feasible solution exists and if so its complexity. In the offline models, we assume that the input traffic is known, available bandwidths is constant equal to the link capacity and intradomain traffic is steady and does not have an affect on the latencies. For the sake of simplicity, in this section we effectively consider only the first two objectives and the third objective about minimising the intradomain link cost is not considered. These models are equivalent to a scenario where we collect the traffic traces before hand and replay them. Hence we call these models the offline models.

5.3.1 Input Traffic Known

To start with let us assume that the input traffic is known. More specifically we assume that the incoming traffic of each of the users (b_i) is known. The sum of these $b_i's$ will yield

the total incoming traffic. Since we know the assignment of users to links, we can also compute the utilisation of each individual link. Further as we are considering a static network the available bandwidth of each link is assumed to be a constant equal to the link capacity. Using equation 5.1, we can compute the ideal utilisation of the egress links. Finally our job will be to distribute the user traffic such that utilisation of each link is at its ideal value. We propose to solve the problem in two steps. First we compute the state of each link and ascertain the amount of traffic that needs to be re-assigned in order to make the links load balanced. In the second step we do the actual user re-assignment. The bandwidth of a link is measured over a period of time. Thus any algorithm that we propose must also proceed in periods (stages).

i. Rank the Egress Links : The utilisation of a link with respect to its ideal value will be in one of the three states - *over-utilised*, *utilised* or *under-utilised*. A link is classified as *over-utilised* if its utilisation is above the ideal value, *utilised* if it is exactly equal and *under-utilised* if it is below the ideal value. We define a rank for each of the links. The rank of a link is the difference in the value between its ideal and actual utilisation. An algorithm to compute the rank of the egress links and to list the over-utilised and under-utilised links is given in algorithm 1. The input to the algorithm is the incoming user traffic, available bandwidth and user assignments. The algorithm computes the rank of all the links. A link is categorised as over-utilised if its rank is negative and under-utilised if it is positive.

ii. Move Users: We now have a set of users assigned to *over-utilised* links from which we need to select some of the users and move them to *under-utilised* links so that the link utilisations become as close as possible to their ideal value. Our goal is to load balance the traffic with a minimum number of user re-allocations. We call this problem the re-assignment problem. We prove that the problem is intractable by showing that a restricted instance of the problem (restricted re-assignment problem) is NP-complete. A formal def-

```
input : b[1..S], A[1..N], x[1..S]
/* b[]:  Incoming User Traffic                                   */
/* A[]:  Link Available Bandwidth                                */
/* x[i]=e, user i assigned link e                                */
for e = 1 to N do
    T_Abw ← T_Abw + A[e]
end
/* U[] Measured Incoming Traffic                                 */
for i = 1 to S do
    T_traffic ← T_traffic + b[i]
    U[x[i]] ← U[x[i]] + b[i]
end
/* IU[] Ideal Utilisation                                        */
for e = 1 to N do
    IU[e] ← (T_traffic . A[e]) / T_Abw
    rank[e] ← IU[e] − U[e]
    if rank[e] < 0 then
        /* OUtil[] Over-utilised links                           */
        OUtil[index++] ← e
    end
    else if rank[e] > 0 then
        /* UUtil[] Under-utilised links                          */
        UUtil[index1++] ← e
    end
end
```

Algorithm 1: Compute Rank of Egress Links

inition of the restricted re-assignment problem is given in definition 1. In theorem 1 we prove that the restricted problem is NP-complete and hence the re-assignment problem is also NP-complete.

Definition 1. *The restricted re-assignment problem (RRP) is defined as follows. In the offline case when input traffic is known, we assume a network scenario where the total number of egress links present is 2. Available bandwidth of both the links is assumed to be equal and known. The ideal utilisation of both the links therefore should be same, equal to half of the total incoming traffic. The problem is to re-allocate some users from the*

over-utilised to the under-utilised link such that utilisation of both the links become ideal.

Theorem 1. *The restricted re-assignment problem is NP-complete.*

Proof. The proof is in two steps. We first show that the problem is in NP and that this reduction can be carried out in polynomial time. Next we show that the problem is equivalent to the partition problem, a well known NP-complete problem. To begin with we define an instance of our problem precisely.

Consider a finite user set I=$\{i_1, .., i_S\}$, incoming traffic of user i is b_i. E =$\{e_u, e_o\}$ is the two element set of egress links, where e_u is an under-utilised link and e_o is an over-utilised link. Available bandwidth of the links is $A_{e_u} = A_{e_o} = A$. The question is can we select 2 disjoint subsets, $I_u, I_o \subseteq I$ where $I_o = I - I_u$ such that

$$\sum_{j \in I_u} b_j = \sum_{k \in I_o} b_k = \frac{\sum_{i=1}^{S} b_i}{2}$$

It is easy to see that the problem is in NP. A non-deterministic algorithm needs to "guess" the 2 subsets; we can then verify the result in polynomial time.

An instance of the partition problem as defined in ([54]) is reproduced below.

A finite set L and a "size" $s(l) \in \mathbf{Z}^+$ for each $l \in L$. The question is does a subset $L' \subseteq L$ exists such that

$$\sum_{l \in L'} s(l) = \sum_{l \in L - L'} s(l).$$

We show that our problem is equivalent to the partition problem as follows:

- The set of users I is the finite set L of the partition problem, $|I| = |L|$. Each $b_i \in I$ is $s(i) \in L$.
- The subset I_u is L' and subset I_o is L-L'.

This reduction can be done in polynomial time and hence the proof. □

Corollary 1. *The re-assignment problem is NP-complete.*

Proof. An instance of the re-assignment problem can be defined as follows.

Consider a finite user set I=$\{i_1, .., i_S\}$, with incoming traffic b_i. E =$\{e_1, \ldots, e_N\}$ is the set of egress links with available bandwidth $A_{e_1} \leq \ldots \leq A_{e_N}$. The question is can we select N disjoint subsets, $I_j \subseteq I$, j=1, $\ldots$, N, such that

$$\sum_{j \in I_j} b_j = \frac{\sum_{i=1}^{S} b_i \ . \ A_{e_j}}{\sum_{k=1}^{N} A_{e_k}} \ \forall j = 1, \ldots, N$$

Clearly, there is a one-to-one correspondence between the re-assignment problem and RRP. If we specify the following two additional restrictions to the re-assignment problem, N=2 and $A_{e_1} = A_{e_2}$, then the resultant problem is the RRP. Hence from proof by restriction ([54]), the re-assignment problem is NP-complete. □

Having shown that our problem is NP-complete, we next establish a correlation between the over-utilised and under-utilised link in theorem 2. The theorem states that if one or more of the egress links are over-utilised then there are guaranteed to be under-utilised link(s) which will exactly fit the excess traffic. The proof of the theorem is simple and directly follows from our definitions. A formal proof is given below. This correlation indicates that feasible solutions to the problem exist if the input traffic can be discretised at appropriate levels. A feasible solution is a set of under-utilised links which can accommodate the excess traffic of the over-utilised links. In the following sections, we explore the possibilities of a solution by introducing additional constraints on the input traffic.

Theorem 2. *If there are over-utilised links then there must exists one or more under-utilised links. The absolute value of sum of ranks of over-utilised links must equal sum of ranks of under-utilised links.*

Proof. Suppose there are N links. Let $IU_e(t)$, $U_e(t)$ and $A_e(t)$ be the ideal utilisation, actual utilisation and available bandwidth of a link of a line e at time t. From our definition of ideal utilisation of a link as given in equation 5.1, it follows that the sum of the ideal utilisation of the links will be equal to the sum of their utilisation.

$$\sum_{e=1}^{e=N} IU_e(t) = K \sum_{e=1}^{e=N} A_e(t) = \frac{\sum_{e=1}^{N} U_e(t)}{\sum_{e=1}^{N} A_e(t)} \sum_{e=1}^{e=N} A_e(t)$$
$$= \sum_{e=1}^{e=N} U_e(t) = \text{Total incoming traffic at time t.}$$

We pair the ideal utilisation and utilisation of a link and rewrite the above equation as follows.

$$(IU_1 - U_1) + (IU_2 - U_2) + \ldots + (IU_N - U_N) = 0 \tag{5.4}$$

Each term in equation 5.4 represents the rank of the corresponding link. Since the sum total of the ranks is zero, it means either all the terms are zero or there are some negative terms and positive terms which cancel out each other. The first possibility that all the ranks are zero is unlikely since it will mean ideal utilisation and utilisation of all the links are the same. In practice utilisation of a link will be either greater or lesser than its ideal value. Suppose utilisation of the links l and m are greater than their ideal value (over-utilised links). It means the terms $(IU_l - U_l)$ and $(IU_m - U_m)$ are negative. Since the net value of the equation must be 0, there will be one or more positive terms (under-utilised links) present in the equation. Without loss of generality, let n be the only under-utilised link. This means the links l, m and n have non-zero ranks. Thus equation 5.4 can be re-written as:

$$(IU_l - U_l) + (IU_m - U_m) + (IU_n - U_n) = 0 \text{ or}$$
$$|(IU_l - U_l) + (IU_m - U_m)| = |(IU_n - U_n)|$$

Hence the proof. □

5.3.2 Input Traffic Known and Bounded.

Theorem 2 suggests that a feasible solution to the re-assignment problem exists if the input traffic can be suitably restrained. We therefore enforce additional restrictions on the input traffic and examine if a polynomial time solution can be realised. The incoming traffic of a user (b_i) cannot exceed the available bandwidth of the link to which it is assigned. Thus, we put an upper bound on the user bandwidth equal to the available bandwidth of the link. The problem is now equivalent to a partition problem where the "size" of the elements of the set are bounded. Garey and Johnson ([53]) show that in the partition problem, *pseudo-polynomial* time bounds solutions are possible if the sum of the "sizes" of the elements of the set are known in advance.

Let us further restrict the values of b_i to two - 0 or 1. A value of 0 means a user is not receiving any traffic (dormant) while 1 means the user is receiving traffic (active). Then from theorem 2, it follows that an exact solution to the problem can be found by filling up each under-utilised link with users from over-utilised link that have value of b_i equal to 1, such that the utilisation of all these links become ideal. All we need to do is to identify such users assigned to over-utilised links and move them one by one. A straight forward approach would be to examine each user, identify the rank of the link assigned to the user, move those users assigned to over-utilised links and then update the link utilisations and ranks. The solution will, therefore, take at most $\odot(S)$ time, where S is the size of the user population.

The conclusion that we can draw from the above discussions is that one may not be able to find an exact solution to the problem even if the input traffic is assumed to be known unless the traffic is suitably discretised.

5.3.3 Input Traffic Known, Objectives Assigned Preferences

In the literature there are a number of methods available to solve multi-objective optimisation problems. In ([92]), Marler and Arora provide a comprehensive review of multi-objective optimisation methods. One such method is the bounded objective function method. The idea is to assign preferences to objectives a priori to convert the problem into a simpler, single objective problem either by aggregating the objective functions or optimizing one objective and treating the others as constraints.

In the context of our problem, movement of users will incur a relatively larger overhead - changing of routing table entries, preserving existing connections and route re-advertisements. Balancing the traffic load is important to prevent congestion. But traffic will vary between periods and in practice exactly balancing the link utilisations will be difficult. A practical approach, therefore, would be to attempt to minimise the user movements while incurring some disparity on load balancing of the links. We rewrite the problem as a linear program with minimisation of user movements considered as the lone objective and the objective to load balance the links as constraints.

$$min \sum_{e=1}^{N} \sum_{i=1}^{S} |x_{ie}(t) - x_{ie}(t-1)|$$

such that

$$\sum_{e=1}^{N} |IU_e - U_e| \leq D, \ \forall e \in E$$

where D is the maximum allowance utilisation of a link can vary with respect to its ideal value. We do not go into details of solving the linear program because we will find that in the online scenario, where the input traffic becomes unknown, assigning of such preferences to objectives does not make the problem any simpler.

5.4 Online Models

The offline analysis was carried out to gain insight into the problem. In the offline analysis we have shown that the problem is NP-complete even if the input traffic is known. A polynomial time solution could be achieved only after imposing several levels of restrictions on the input traffic. In the real world input traffic will change dynamically and their values will be unknown at the time of assigning users to links. The algorithms that we propose in this section immediately process the input traffic as it enters the network, thus we refer to these network models as the online models. Since in the online model the input traffic is assumed to be unknown, it is unlikely that we can have a simpler solution than that of the offline models. However, we continue with what we have learned and explore possibilities of adapting these solutions to the online scenario.

5.4.1 Input Traffic Unknown, Objectives Assigned Preference and Intradomain Traffic Static

Stochastic linear programming is a framework to handle uncertainty in linear programs by modeling the uncertainty in the problem data as a probability distribution. In general terms, the assignment problem consists of finding the best assignment of tasks to agents according to a predefined objective function. However, in practice each agent requires a quantity of some limited resource to process a given job. The problem of assignments taking into account these resource constraints is known as the Generalised Assignment Problem (GAP). The assignments of jobs to agents must be done before the actual demands for resource capacity are known. In the recourse model, once the values become known, it is possible to re-assign some of the jobs incurring costs. The goal is to minimise the re-assignment costs. It comes in two flavors - two-stage and multi-stage recourse model. Two-stage stochastic recourse models allow decisions to be taken in two stages - first before the experiment and a corrective decision (recourse) once the uncertainty is revealed after the experiment. Decisions taken before the experiment are called first-stage decisions and

those after the experiment are called second stage decisions. In a multi-stage stochastic program with recourse the uncertainty in the problem evolves through a series of stages. At each stage the uncertainty is partially revealed and one takes decision in each stage in response to the uncertainty realised so far. The uncertainty is fully revealed at the end of the period and the sequence of the decisions is to optimize the uncertainty revealed at the end of the period. Our problem has characteristics of both the recourse models but does not exactly fit into either of them. The distribution of the stochastic parameter (input traffic) at each stage is independent which makes it appear as a two-stage recourse model. However, like the multi-stage model, the distribution of the stochastic parameter gets partially revealed at each stage. Continuing with our linear programming approach in the offline analysis, we propose to model our problem as a *recursive* two-stage model. We define a stage as equivalent to a period. A formal representation of a two-stage recursive model is given below:

$$min\ z\ =\ c^T x\ +\ \mathbf{E}_\xi[\mathbb{Q}(x, \xi(\omega))] \tag{5.5}$$

where

$$\mathbb{Q}(x, \xi(\omega)) = min[q(\omega)^T y(\omega)]$$

In the beginning of the first stage users are provisionally assigned links. The first term in equation 5.5 represents the first stage direct cost where x is the assignment of users and c is the cost vector. Depending on the input traffic realised in the first stage the algorithm will take a recourse action, which is re-assign some of the users. The second term represents the expected cost of re-assignment. $\mathbb{Q}$ is the recourse function, q is a vector of recourse costs or the penalties for re-assignment. $\xi(\omega))$ is the expected number of recourses and $y(\omega)$ is the actual recourse actions taken which is equivalent to re-assignments in the context of our problem. The second stage determines the final assignment plan once the uncertainties are known. The aim is to minimise the recourse cost. The second stage will act as the

first stage for the next run of the algorithm and so on the algorithm will proceeds recursively.

Stochastic programs are, however, difficult to solve and a two-stage recourse problem has been shown to be a #P-hard problem ([58]). The difficulty in solving the stochastic linear program will mainly arise because the distribution of the random parameter (input traffic) is unknown and the objective function is a multivariate expectation, which cannot be computed exactly. No results on exact algorithms for any type of stochastic recourse models are known. The most commonly used approach to tackle stochastic linear programming problem is to use approximation methods. The main disadvantage of formulating the problem as a stochastic linear programming model is that it requires assigning preferences to the objectives, thereby diluting our goal. But, this act of restricting our goal does not make the problem any simpler.

5.4.2 Input Traffic Unknown, Intradomain Traffic Static

In the offline analysis we have shown that the problem is NP-complete even in the simple case when the input traffic is assumed to be known. Further we have shown that the act of imposing restrictions on the objectives does not make the problem any simpler. We now remove all the restrictions imposed on the objectives and propose objectives. The input traffic is assumed to be unknown and can take on any value. However, we assume that the intradomain traffic is steady and does not have an affect on the latencies. We call such a network environment where the effect of intradomain traffic is not considered as a *static network environment* or a *static intradomain network environment*. The network traffic is not entirely static since the incoming traffic is dynamic and available bandwidth of the egress links can vary. One may comprehend that the state of the network assumed in this section is similar to that of the LAN-based model we discussed in the preceding chapter.

Greedy Approach

Ideally, the problem of user re-assignment should be considered at the global level by a *bandwidth manager*. The manager should be able to see the traffic dynamics of the whole network as well as the current user assignments and then re-assign the users. The proposed heuristics are broadly structured as the two-stage recourse model. The heuristic proceeds in stages or periods. Duration of a period is set as before to 5 minutes. At the end of each stage, the utilisations and ranks of the links are computed. Based on the current network state, we compute the minimum number of user re-assignments that will make the utilisations of the links as close as possible to their ideal value. The issues that need to be addressed are - (i) how many users are to be re-assigned (ii) which are the candidate users for re-assignment and (iii) in case of more then one under-utilised link to which of these links do we move the user. Our first heuristic is to move the user that is currently receiving the maximum traffic to the most under-utilised link. We call this heuristic the greedy approach. The actions that need to be performed at the end of each period are summarised below:

i. Compute rank of the links based on the current network traffic state using algorithm 1.

ii. Sort the users assigned to over-utilised links in descending order of their $b_i's$ (incoming traffic) and consider them one by one.

iii. Select the first user from the sorted list. Check if the user can be moved. Get the most under-utilised link. If the user *fits* the under-utilised go to the next step. Else repeat (iii) till a user can be moved; exit if no more users are left in the sorted list.

iv. Move the user and re-compute the link utilisations and ranks.

v. Repeat steps (iii) and (iv) until utilisation of all the links become optimal or no more users can be moved.

When a user is selected for re-assignment, the algorithm first checks if moving the user will make the over-utilised link under-utilised. Next the algorithm checks if the user *fits* the selected under-utilised link. We say that a user *fits* into a link if assigning the user to the link does not make the link over-utilised. This checking is to ensure that due to our user re-assignments a currently under-utilised link does not become over-utilised. Since the link ranks are re-computed after every user re-assignment, the algorithm will halt under two circumstances. In the first case when utilisation of all the links become ideal. Incoming traffic of a user can vary over a wide range. Therefore, the more likely situation when the algorithm will halt is that there is no link that *fits* a user. A pseudo-code of this heuristic is given in algorithm 2.

Random User Selection Approach

Depending on the manner we select users for re-assignment, there can be different algorithms. Internet traffic variability, that is the number of sources that become active or stop sending traffic, is high for smaller time scales ([99]). Stated plainly, this means the present highest traffic receiver may not remain so in the next few periods. Therefore, in the second heuristic we do not discriminate the users on the basis of their current bandwidth usage. Rather the users are selected based on the outcome of a random game. We call this approach the Random User Selection (RUS) approach. Given an over-utilised link, we compute its *relative utilisation*, that is utilisation of the link over its available bandwidth. All users assigned to the over-utilised link are then moved with a probability equal to the *relative utilisation* of the link to the most under-utilised link. The higher the *relative utilisation* of an over-utilised link, the greater is the probability that more users will be moved from the link. Since utilisation of the links gets updated along with the movement of users,

```
input : OUtil[], UUtil[]
/* Definition of input variables same as in algorithm 1.            */
/* OUser[]:  Users assigned to over-utilised links.                 */
for i = 1 to S do
    if (rank[x[i]] < 0) then
        OUser[index++] ← i
    end
end
/* Sort users in descending order.                                  */
SortedOuser ← sort(OUser,"d")
foreach ou in SortedOUser do
    lnk ← x[ou]
    /* Check if moving user ou will make link "lnk" under-utilised  */
    tmp_util_lnk ← U[lnk] − b[ou]
    tmp_rank_lnk ← IU[lnk] − tmp_util_lnk
    if (tmp_rank_lnk ≤ 0) then
        /* Get most under-utilised link, getnext defined in algo. 3.
           */
        next ← getnext()
        /* Check if user ou fits the under-utilised link "next"     */
        tmp_util_next ← U[next] + b[ou]
        tmp_rank_next ← IU[next] − tmp_util_next
        if (tmp_rank_next ≥ 0) then
            /* Move user; update link utilisation and ranks.        */
            x[ou] ← next
            U[lnk] ← tmp_util_lnk
            rank[lnk] ← tmp_rank_lnk
            U[next] ← tmp_util_next
            rank[next] ← tmp_rank_next
        end
    end
end
```

Algorithm 2: Greedy Algorithm

```
getnext()
/* Return most under-utilised link.                                    */
max_rank ← 0
ret_lnk ← −1
foreach lnk in UUtil do
    if rank[lnk] > max_rank then
        max_rank ← rank[lnk]
        ret_lnk ← lnk
    end
end
return ret_lnk
```

Algorithm 3: getnext() Function

relative utilisation of the links will also change accordingly. The algorithm will halt under the same situation as in the case of the greedy approach. The difference is that in the RUS approach, a user that fits an under-utilised may not get moved and the algorithm may eventually run out of users. Nevertheless, both the heuristics move traffic so that the utilisation of the links becomes as close as possible to their ideal value. Thus if there are sufficient number of users available, the performance of both the greedy and random approach with respect to link utilisations will be comparable. The main difference between these two heuristics will be in the number of users re-assigned. A pseudo-code of the RUS heuristic is given in algorithm 4, it has a run-time of $\mathsf{O}(S)$ as compared to the $\mathsf{O}(SlogS)$ of the greedy approach. The *getnext* function is the same for both the algorithm.

5.4.3 Input Traffic Unknown, Network State Dynamic

So far we have deliberated situations where the input traffic and available bandwidth can vary. Now we finally assume a totally dynamic network environment where the input Internet traffic is unknown, available bandwidth varies across periods and the intradomain traffic is dynamic too and can have an affect on the latencies. In a sense both the algorithms we have proposed for the static network environment is greedy in nature since it selects the most under-utilised link. Another possibility of selecting an under-utilised link would be to select one that best fits the user. Again we can have a number of heuristics

```
input : OUtil[], UUtil[], b[1,..,S], rank[1,..N], x[1,..S], u[1..N], A[1..N]
/* Definition of input variables same as in algorithm 1.                    */
/* OUser[]:  List of users assigned to over-utilised links.                 */
for i = 1 to S do
    if (rank[x[i]] < 0) then
        OUser[index++] ← i
    end
end
foreach ou in OUser do
    lnk ← x[ou]
    /* gen():  Generate a random number using uniform distribution.         */
    prob ← gen (UD)
    /* Compute the relative Utilisation                                     */
    rel_util ← u[lnk] / A[lnk]
    if (prob < rel_util) then
        /* Check if moving user ou makes link "lnk" under-utilised          */
        tmp_util_lnk ← U[lnk] − b[ou]
        tmp_rank_lnk ← IU[lnk] − tmp_util_lnk
        if (tmp_rank_lnk ≤ 0) then
            /* Get a under-utilised link, getnext defined in algo.  3.
               */
            next ← getnext()
            /* Check if user ou fits the under-utilised link "next"         */
            tmp_util_next ← U[next] + b[ou]
            tmp_rank_next ← IU[next] − tmp_util_next
            if (tmp_rank_next ≥ 0) then
                /* Move user; update link utilisation and ranks.            */
                x[ou] ← next
                U[lnk] ← tmp_util_lnk
                rank[lnk] ← tmp_rank_lnk
                U[next] ← tmp_util_next
                rank[next] ← tmp_rank_next
            end
        end
    end
end
```

Algorithm 4: Random User Selection Algorithm

depending on the way we select an under-utilised link. In this section our aim is to select an under-utilised link such that the dynamics of intradomain traffic are taken into account. To deal with the problem, let us first study how existing Internet protocols handle such situations. In BGP, if the route selection process results in multiple equally good egress routes then the intradomain distance is used to break the tie. The route with the smallest IGP distance is selected from a set of equally good egress routes. We propose to follow a similar philosophy in our work. For a user assigned to an over-utilised link, the set of under-utilised links that *fits* the user are equally good egress routes. Taking a cue from BGP, from the set of under-utilised links we select one with the least intradomain cost. To incorporate the above alterations into our heuristics, we need to only change the way an under-utilised link is selected for moving users. As such, we shall continue to use the same strategies to select a user for re-assignment, that is algorithm 2 and algorithm 4 will remain unchanged. The only change required is in the *getnext* function. Instead of selecting the most under-utilised link, for a given user we compute the IGP distance from the user to the egress router of all the under-utilised links and select the one with the least cost. A pseudo-code of the new getnext function is given in algorithm 5.

```
getnext(user)
/* Return least cost link.                                    */
src ← user
min_cost ← 999
ret_lnk ← −1
foreach lnk in UUtil do
    dst ← lnk
    /* Cost between src and dst                               */
    cost ← pathcost(src, dst)
    if cost < min_cost then
        min_cost ← cost
        ret_lnk ← lnk
    end
end
return ret_lnk
```

Algorithm 5: Returns the least IGP cost under-utilised link

5.4.4 Forwarding Traffic to Egress Routes

Intradomain routing protocols traditionally use IGP metrics to choose egress route of users. Our algorithms on the other hand demand that the user traffic be directed to their specific assigned egress routes. One way of achieving this would be to forward traffic of all users to a centralised manger which in turn will route the user traffic. This is the forwarding mechanism we have used in our threshold-based route control scheme proposed in chapter 4. Such an *in-band* forwarding mechanism although simple is not scalable and in larger networks the manager will become a major source of bottleneck. The availability of a large number of tunneling technologies allows for more sophisticated forwarding mechanism. Tunneling technologies like Multi Protocol Label Switching ([45]) allow providers greater flexibility in network-deployment choices, improved routing system scalability, and greater reach to customers. We can take the advantage of such technologies to create label switched paths (LSPs) between a user and its egress route and then forward the traffic *out-of-band*. With such a mechanism the centralised manager only needs to pass on information of the egress router assigned to users thereby greatly reducing the load on the manager.

5.5 Experimental Evaluation

In this section we describe our experimental evaluation of the various route control alternatives proposed in section 5.4. These include evaluating the performance of the route control models for static intradomain network environment and examining the sensitivity of the model to cross-traffic. We concentrate on highlighting the performance benefits of each scheme. As we proceed with our experimental analysis, based on the output of our experiments we propose modification to our heuristics to maximise benefit. In the final part of our evaluation we generate a dynamic network scenario so as to test our dynamic route control algorithms in a realistic network setting. The algorithms are tested with both synthetic and actual traffic traces. The algorithms are tested using a wide range of possible

scenarios and exhaustive data sets.

5.5.1 Static Network Environment

In this section we test the static route control algorithms. The setup for the static network environment is simple. The performance of our proposed heuristics is evaluated using a simulation program written in Perl. The simulations are performed using actual traffic traces. Although the simulation program does not consider intradomain traffic dynamics, nevertheless we can get a feel of the performance of the algorithms in actual traffic conditions. To quantify the benefits of our approach we compare our heuristics to the *default* case when no load balancing is applied and measure the improvements in performance.

Experiment Setup

The packet sniffer program tcpdump ([12]) is used to collect network traffic traces from the different egress links. Each individual trace represents the utilisation of a link. The traces are initially analysed individually. As discussed in section 5.1.2, we regard flows as a set of packets that travel in either direction between a pair of communication endpoints. We basically consider two flows - incoming and outgoing flows. Time-stamp of the first packet marks the beginning of an experiment. Payloads of all incoming traffic flows that fall within a period are added up. This sum represents the *default* utilisation of the links when no route control techniques are used to load balance the traffic. While computing the link utilisations, we also extract the distinct users (IP addresses) present in the trace.

In the second part of our analysis we try to emulate the behavior of the centralised manager, who can see the traffic of all the egress routes as well as the user assignments. The individual traces are merged in chronological order using tcpslice ([13]), a tool to glue together tcpdump files. The merged trace acts as the input to the simulation program. Since we know the users present in each of the individual traces, it means we also know the

initial or default assignment of users to egress links. The program computes the utilisation and rank of all the egress links in a period. Users assigned to over-utilised links are moved out to under-utilised links. The selection of users and under-utilised links depend on the algorithm simulated. For example if we are simulating the greedy algorithm, the users are sorted in decreasing order of their size and then they are moved out one at a time until the utilisation of the links become ideal or there are no users that can be moved. In this work, by size we mean the total incoming traffic of a user in a given time duration.

Data Set

Experimental results will largely depend on the characteristics of the traffic traces used. Traces used in our interdomain traffic engineering experiments must have certain properties. First, the traces should be collected at transit points so that individual traffic from all the users can be seen. Secondly the network should be multihomed and traces should ideally cover traffic on at least two different interdomain links. We achieve this by collecting individual trace from the different interdomain links simultaneously and then merge them. Thirdly we should be able to construct from the traces, the source and destination addresses, volume of traffic and IP protocol. We consider only TCP connections since TCP constitutes bulk of the Internet traffic.

The traces used in the experiments were collected at our institute's transit points. The institute has 48 Mbps of Internet bandwidth subscribed from 5 different ISPs. Traces were collected from each of the egress links for different durations and on different days of the week. Though we don't intent to study interdomain traffic in this work, we examine the traces to show that our data set possess the basic interdomain traffic characteristics ([108], [33]). In this section we present our observations from the analysis of a 1 hour trace collected from 4 different ISP links. The total number of TCP connections detected in the trace was 22.7 millions and the distinct number of users (IP addresses) present was 932.

The total traffic downloaded was 7.5 gigabytes and outgoing traffic was 0.5 gigabytes. The highest traffic downloaded by a user was 607 megabytes and the user was active throughout the duration of the trace period. On the other hand the lowest size user downloaded just 67 bytes and it was active for about 1 sec only. Traffic flows are usually classified on the basis of their size and lifetime. We analysed the trace both in terms of the bytes downloaded and lifetime of a user. The data set exhibit all the characteristics of interdomain traffic. Results of our study is given below.

- We sorted the users on the basis of their size and plot the cumulative percentage of bytes downloaded in figure 5.2. A small fraction of users account for the major volume of traffic. As can be seen, 5 percent of the users account for 70 percent of the traffic and 10 percent of the users account for 80 percent of the traffic. These users have high throughput and long duration flows (elephants).

- Next we classified the users on the basis of their lifetime and placed them in bins of 5 minutes. The percentage of users present in each bin and the total bytes downloaded is plotted as a histogram in figure 5.3. The choice of 5 minutes bins is obvious; our proposed algorithms execute recursively in periods of the same duration. Out of the 12 bins, 40 percent of the users lie in the first bin. However, the total bytes downloaded by these users are just 4.5 percent. That is the majority of the users are short-lived having a lifetime of less than 5 minutes and have a low throughput (mice). Such users will not have much of an impact on load balancing of the links. The percentage of users present in a bin almost linearly decreases as the bins move further away from the y-axis. This means very few users have long duration lifetime.

- There does not seem to be any correlation between the size and lifetime of a user. Earlier studies ([33]) have also indicated that flow size is independent of the flow lifetime.

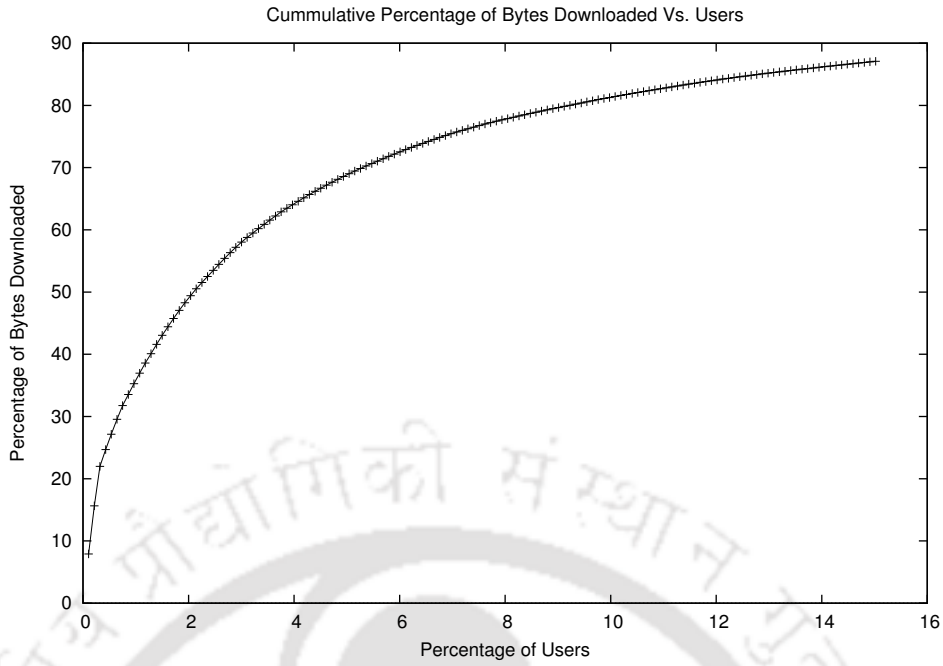


Fig. 5.2 Bytes downloaded as a function of users.

Experiment 1

In this experiment we considered traces collected from three different egress links. The duration of each trace was 90 minutes and the number of distinct users (IP addresses) present was 600. The traces were collected simultaneously from all the three links. We computed the utilisation of the individual links using the traces. The utilisation so measured was for the *default* case, when no load balancing heuristics were used. Next we merged the traces and simulated the greedy approach and the random approach (RUS) on this merged trace.

In figure 5.4 and 5.5 we plot the link utilisations for the three scenarios - the *default* approach, load balancing with greedy and random approach. The available bandwidth of all the three links was considered to be constant, equal to the link capacity. Further the capacities of all the links were considered to be equal. This means that at any instant the utilisations of all the links should be ideally equal. The plot of the *default* case reveals that the utilisations of the links were highly imbalanced. The application of our load balancing

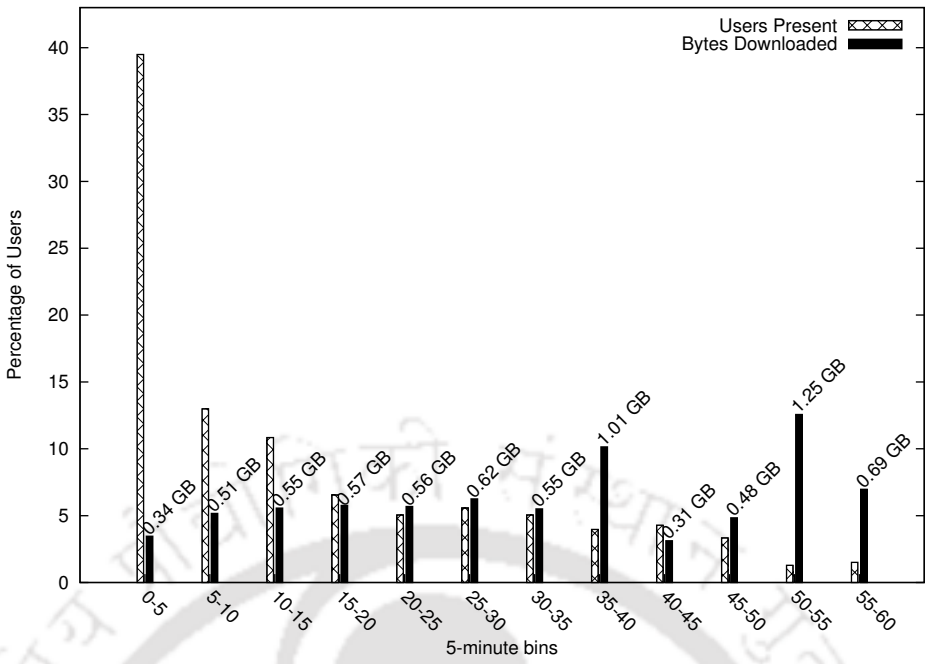


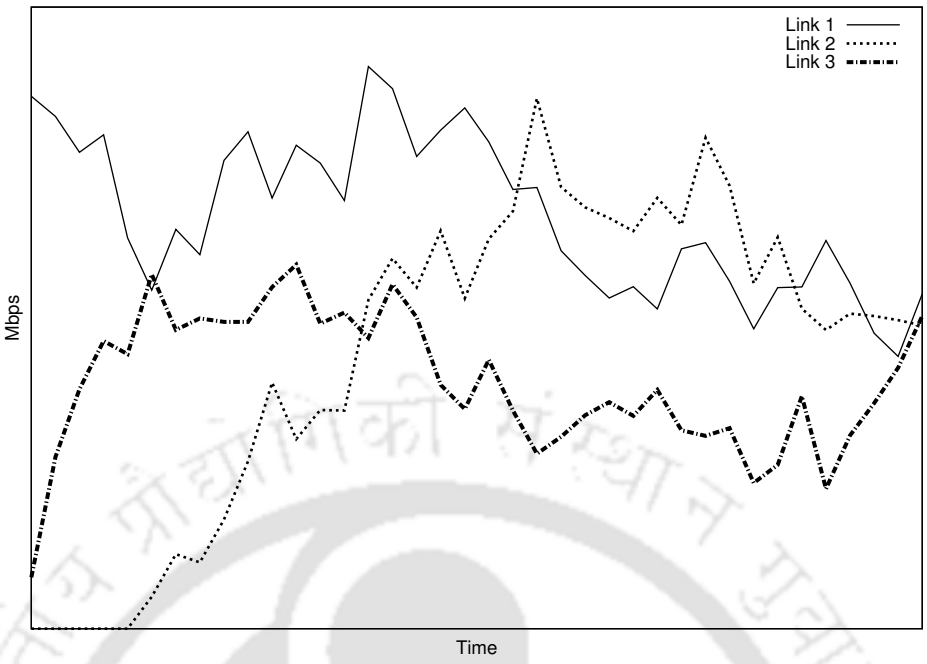
Fig. 5.3 Flow lifetime and bytes downloaded for 5 min. intervals.

heuristics reduced these inequalities in link utilisations to a great extent. The rank of a link is a measure of the deviation of the link's utilisation from the ideal value. To further substantiate our results we tabulate the absolute sum of ranks over all the periods in table 5.1. As expected, the deviation for the *default* case was relatively large, while the values for the greedy and RUS approach were comparable. The application of our load balancing techniques resulted in more than 50 percent better balancing of traffic as compared to the *default* approach. The results of our experiment supports our earlier conjecture that utilisation of the links will be similar for both the greedy and random approach. Finally we compare the number of user movements involved for the greedy and RUS approaches in figure 5.6. The number of user re-assignments for both the heuristics was low, less than 2 percent on an average. From the plots it is evident that the application of our heuristics resulted in better load balancing of the incoming traffic. The random approach is computationally inexpensive as compared to the greedy approach but the price we may have to pay is a larger re-assignment of users. In our experiments we found that employing the RUS approach for load balancing resulted in about 1 percent more user re-assignments in

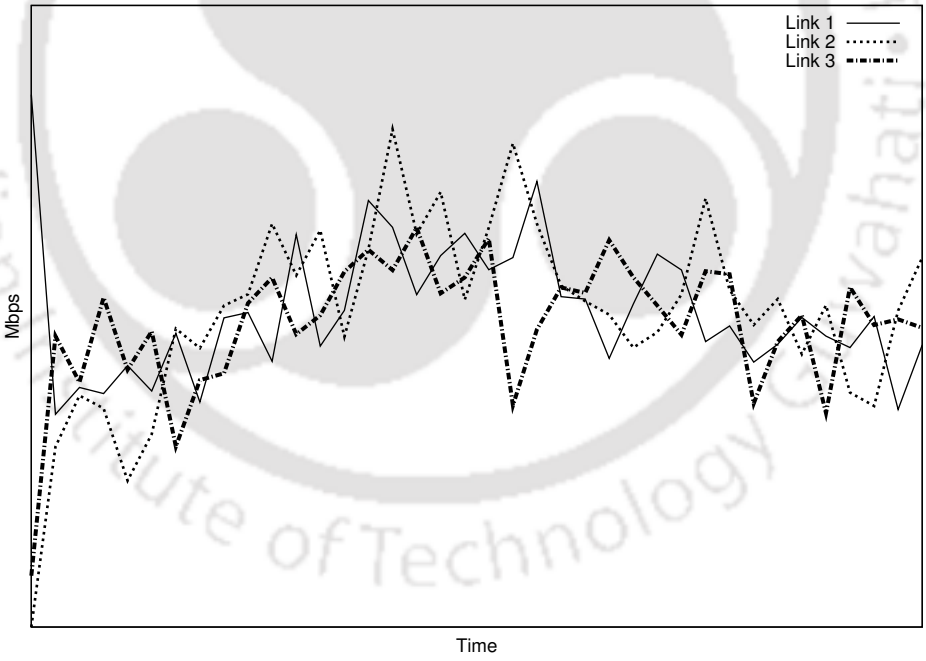
comparison with the greedy approach. In the following experiments we continued with only the greedy approach to further investigate the effects of network traffic on our heuristics.

Table 5.1 Absolute sum of ranks (Mbps)

Default	Greedy	RUS
35.7	13.9	16.74

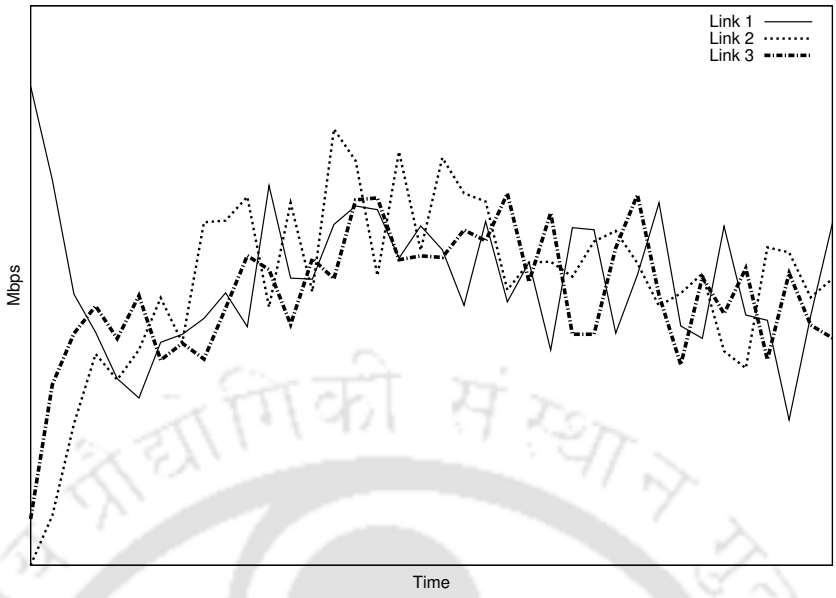


(a) Default



(b) Greedy

Fig. 5.4 Expt. 1: Comparison of link utilisation where all links have same available bandwidth (contd.).



(a) RUS

Fig. 5.5 Expt. 1: Comparison of link utilisation where all links have same available bandwidth.

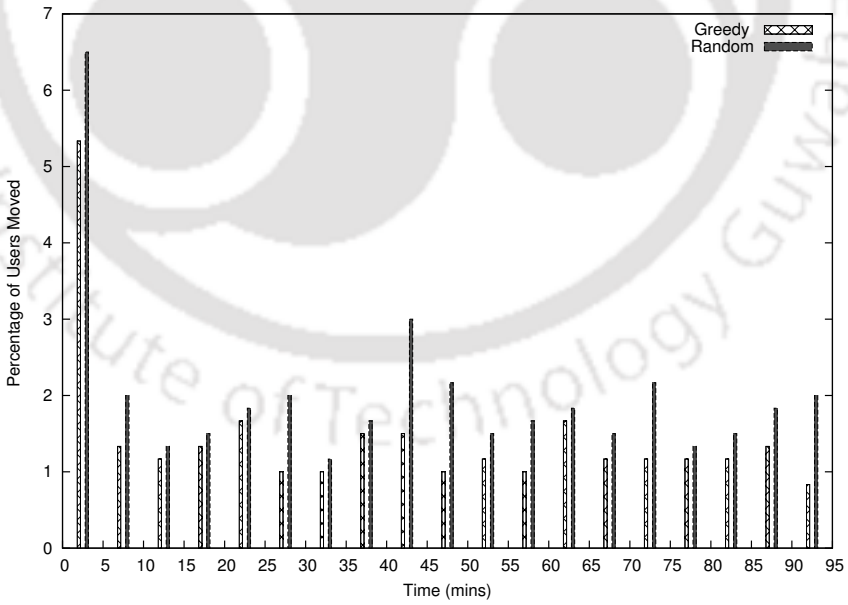


Fig. 5.6 Expt. 1: Comparison of user movements.

Experiment 2: Effect of elephants and mice

The results of our first experiment were very encouraging as we could extract a substantial improvement in performance with minimal user re-assignment. With an aim to test our techniques more rigorously under different traffic scenarios, in this experiment we investigated the effects of the presence of elephants and mice in the traffic. The number of links considered was two and as was in the case of the previous experiment, available bandwidth of both the links was assumed to be constant, equal to their capacities. Duration of the trace collected was 2 hours and the numbers of users present in the trace was 500. As the number of links considered was only two either one of them will be over-utilised and the other under-utilised or utilisation of both the links will be same, which is unlikely. In figure 5.7(a) and 5.7(b) we compare the utilisation of the over-utilised link with its ideal value for the *default* case and greedy approach respectively. The plot of the link utilisation when the greedy algorithm is used looks like a saw tooth. On further investigation, we discovered that this behavior is because there are a few elephants that move to and fro between the two links. When such a user gets moved, the second link becomes over-utilised and the first one under-utilised. In the next period the user gets moved back to the first link and in this way the user oscillates between the two links. This is an example of the *ping-pong* effect we discussed in chapter 4 (section 4.4.1).

To confront such user oscillations, we incorporated an additional constraint to our heuristic. If a user has been moved once it should not be considered for re-assignment for the next few periods. The question is how long should we restrain a user from re-assigned once the user has been moved. To test the limit up to which we can delay re-assignment of users, in our experiments once a user is moved it is not considered for re-assignment during the period of the simulation. This modification leads to significant performance improvement (figure 5.8(a)). Such a move while tackling the issue of user oscillations also has a positive side-effect on route convergence. Route convergence in BGP can range from

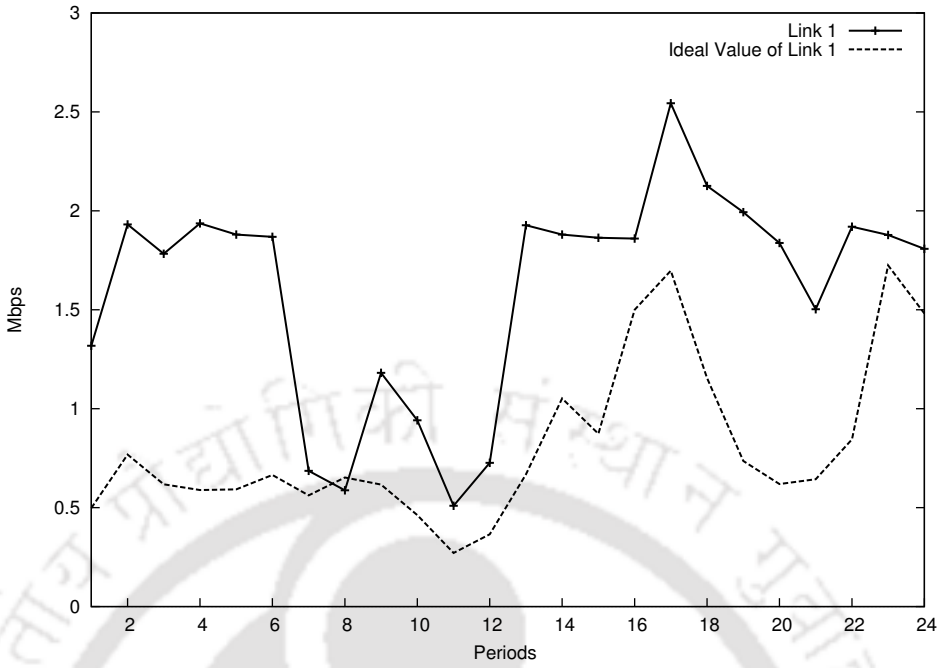
a few minutes to a few hours. After a user is re-assigned if the user is restricted from being re-assigned again for a sufficiently long period, it will ensure that a user does not get moved before its route updates have converged. In practice a domain will have a large user base and this restriction will not have an affect on the load balancing of the links. For example in this experiment even with a relatively limited user base we restrained a user from being re-assigned for 2 hours once it is moved. The performance of the experiment has been satisfactory.

5.5.2 Relaxed Greedy Approach

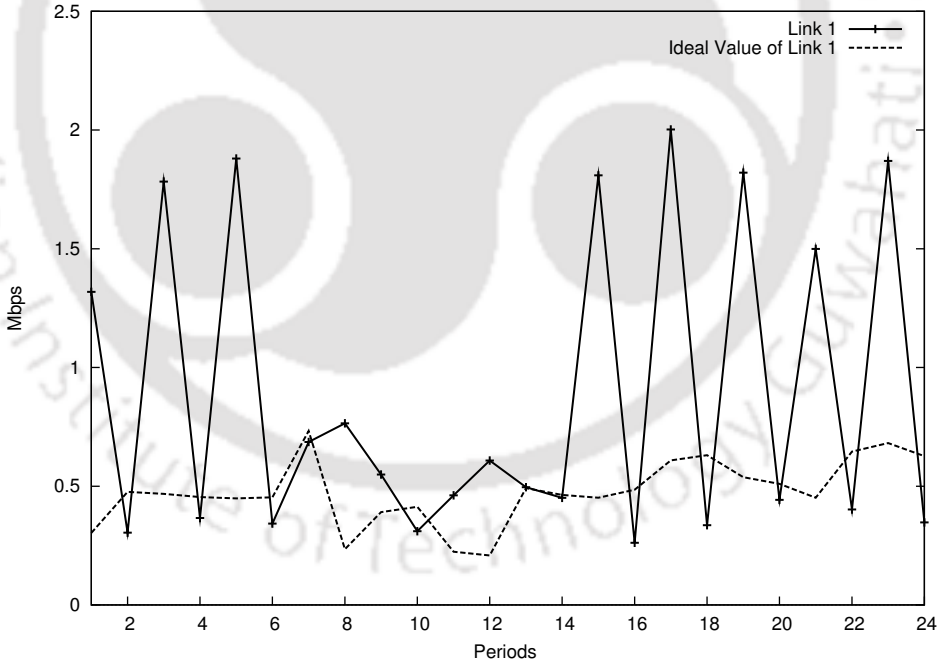
Figure 5.8(a) shows that preventing user oscillations does result in improvement in performance of the greedy approach but there are still periods when the utilisation of the link deviates from its ideal value by about 20 percent. Although this value is not exorbitant there appears to be room for improvement. In our simulations we experimented by moving different numbers and type of users. In this section we report a particular order of user re-assignment that resulted in performance improvement both in terms of link utilisation as well as user re-assignments. In our *original* proposed route control strategies a user is moved to an under-utilised only if the user *fits* the link. Especially in the case of greedy approach, we noticed that in the sorted list of over-utilised users an elephant is usually followed by a large number of mice. An elephant may not get moved because it slightly over-fits the under-utilised link. As a result a large number of mice get moved. The test of whether a user *fits* an under-utilised link has been devised since we do not want an under-utilised link to become over-utilised due to our user re-assignments. In reality utilisation of a link does not depend on the user assignments alone. To an extent link utilisations will also depend on the pattern of traffic in the following period which in any case cannot be predicted. To reduce the number of user movements, we make a second adjustment to our heuristic. Users that slightly over-fit an under-utilised link are allowed to be moved. In order to bring a distinction with our earlier greedy algorithm we call this revised greedy

heuristic the *relaxed greedy* algorithm. We refer to our earlier approach as *strict greedy* or simply greedy algorithm.

An immediate fallout of this relaxation is the issue of how much over-fit do we allow. In this particular experiment the greedy heuristic has been altered such that once a user is selected for re-assignment, it is moved to an under-utilised link without making any other checks. We find that the relaxed greedy approach leads to better load balancing, figure 5.8(b). As can be seen the utilisation plot of the over-utilised link almost coincides with its ideal value. At the same time the number of user movements is significantly reduced. The clustered histogram in figure 5.9 compare the percentage of users moved for the two greedy approaches. The lower performance of the strict greedy approach could be either because sufficient number of low throughput users is not available or due to the short-lived connections of the re-assigned mice. The other question is will the *relaxed greedy* approach always give better performance? During the course of our experiment we find that performance of the strict and relaxed greedy approach depends on the pattern of the input traffic as well as on the network traffic load.

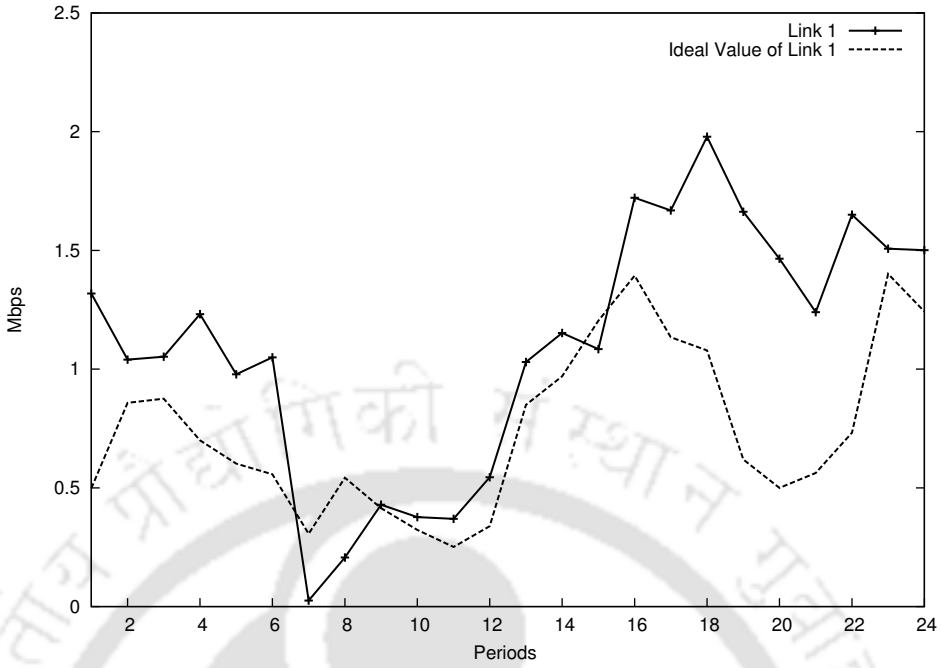


(a) Default, Static route control

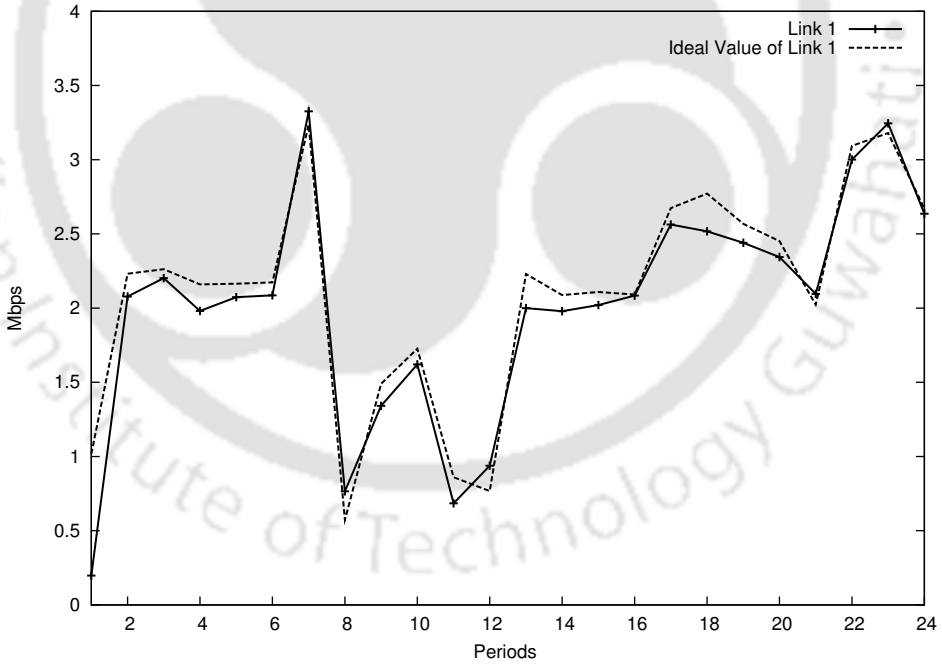


(b) Effect of elephants on greedy approach

Fig. 5.7 Expt. 2: Comparison of link utilisation with their ideal value.



(a) Strict greedy approach



(b) Relaxed greedy approach

Fig. 5.8 Expt. 2: Comparison of link utilisation with their ideal value.

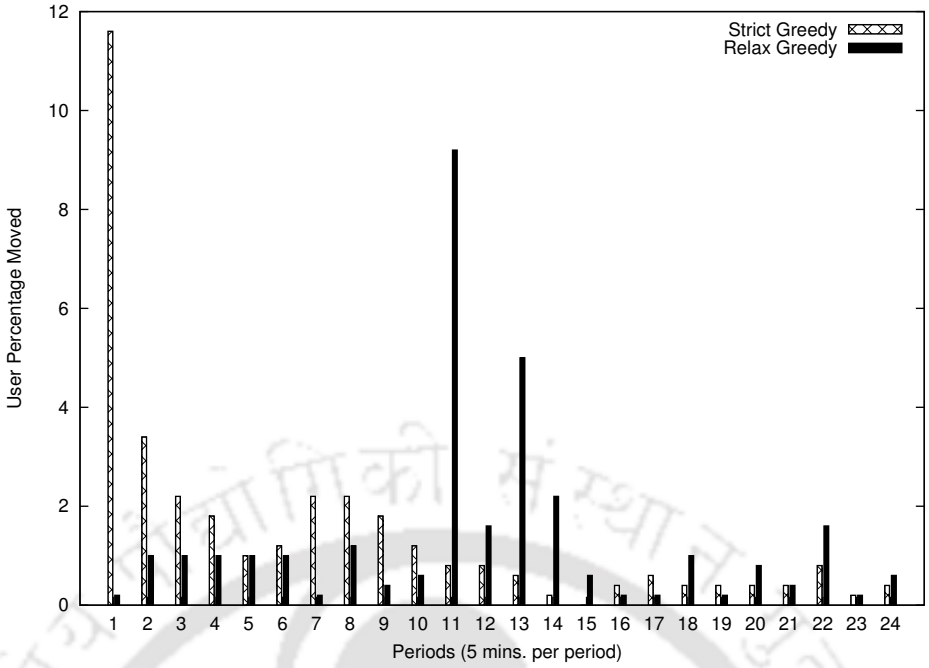


Fig. 5.9 Expt. 2: Percentage of user re-assigned.

5.5.3 Available Bandwidth Varies

In the experiments that we have considered so far available bandwidth of the links was assumed to be a constant. In actual networks available bandwidth of a link will depend on the amount of cross traffic present and it will vary substantially between periods. In order to consider the cross traffic in our simulation, while the traces were collected we also recorded available bandwidth of the egress links using the tools we have previously mentioned in chapter 3.

Experiment 3: Examining the Effects of Cross-Traffic

This particular experiment is to examine the presence of cross traffic on our route control strategies. The number of egress links considered was four, the trace duration was of one hour and the number of users present was 932. The total traffic downloaded was 7.5 gigabytes and the outgoing traffic was 0.5 gigabytes. Both the greedy approaches - strict and relaxed, were simulated using these traces. In order to represent the traffic load bal-

ancing of the entire network, we computed the overall percentage deviation of the greedy approaches; that is the sum of absolute value of all the link ranks divided by the sum of their available bandwidth. The average percentage deviation for the *default*, strict greedy and relax greedy were 18.85, 8.89 and 9.42 respectively. These values indicate that the overall improvement in load balancing as compared to the *default* scenario has been more than 50 percent. However, the values do not show the period-wise improvement in the load balancing performance.

To quantify the improvement period-wise, percentage deviation of the two greedy approaches were compared with that of the *default* case for each period and the difference in value was represented using positive and negative bars. A positive bar shows the quantum of improvement in the deviation of the greedy heuristic with respect to that of the *default* approach. Similarly a negative bar shows the decline in the deviation of the greedy approach from that of the *default* approach. In figure 5.10(a) and 5.10(b) we plot the period-wise deviations of the strict and relax greedy approach respectively. In this run cross-traffic was not considered. As can be seen both the plots have only positive deviations suggesting that there has been an improvement in traffic load balancing in all the periods. The improvement in load balancing as compared to the *default* scenario is more than 50 percent. Further the lengths of the bars are identical indicating that the quantum of benefit is almost same for the entire duration of the simulation. The mean, median and standard deviation of the deviations are given in table 5.2. A positive value indicates positive deviation and a negative value indicates deviation. These values also indicate that the distribution of the positive deviations is largely symmetric. In the case of users re-assignments too, the results are similar. The number of re-assignments are uniform in all the periods. The mean, median and standard deviation of the re-assignments are given in table 5.3. Further like in the previous experiments, we find that the number of users re-assigned is higher for the strict greedy approach.

In the second run of our experiment, we simulated cross traffic by varying available bandwidth of the links in each period as per our recorded values. In this run the deviation of the link utilisations from their ideal value for the *default* scenario is 17.1 percent. The performances of the heuristics marginally degrade (figure 5.11(a) and 5.11(b)) as compared to the previous run when the available bandwidth was considered to be constant. However, most of the deviations are still positive, indicating that the application of our heuristics resulted in better overall performance benefits. The overall improvement in load balancing as compared to the *default* case is more than 40 percent. Table 5.2 further confirm that the performance improvements are almost uniform in all the periods.

Table 5.2 Improvement of route control scheme in comparison to *default* case (in percent)

Route Control	Mean	Median	Standard Deviation
No Cross Traffic			
Strict Greedy	9.96	9.19	3.78
Relax Greedy	9.43	10.2	3.54
Cross Traffic			
Strict Greedy	7.34	7.09	5.95
Relax Greedy	8.61	8.76	3.96
Heavy Cross Traffic			
Strict Greedy	0.9	-0.36	3.78
Relax Greedy	-2.2	-2.27	2.9

When there is no cross traffic present, available bandwidth of the links will entirely depend on the user traffic. Thus the link utilisations will be relatively stable. Moreover, when the available network resources are sufficient, performance will also not significantly degrade even when traffic on the links is not exactly balanced. When cross traffic is present, available bandwidth of the links will depend both on the user traffic as well as cross traffic. Fluctuations in link utilisation will be higher and as a result load balancing of the links will become more difficult. When the cross traffic load is high available bandwidth of the links will become minimal. Although user traffic remains the same, due to non-availability

Table 5.3 User Re-assignments (in percent)

Route Control	Mean	Median	Standard Deviation
No Cross Traffic			
Strict Greedy	1.76	1.3	1.83
Relax Greedy	0.3	0.3	0.28
Cross Traffic			
Strict Greedy	0.65	0.3	1.0
Relax Greedy	0.22	0.2	.08
Heavy Cross Traffic			
Strict Greedy	1.01	1.1	0.4
Relax Greedy	0.22	0.2	0.13

of network resources we can relate such a situation to a scenario where utilisation of all the links has reached their threshold value. To highlight the impact of cross-traffic, in our experiments we simulated an increase in cross-traffic by reducing available bandwidth of the links. As we kept on increasing the cross traffic, the performance degraded and after a certain point the heuristics tend to have a negative impact on the performance (figure 5.12(a) and 5.12(b)). The deviation of the link utilisations from their ideal value for the *default* scenario, in this case is reduced to just 6.47 percent. In a highly dynamic network environment where available bandwidth is small, there will be little room for maneuvering. Moreover, in such situations, the utilisation of a link may fluctuate from over-utilisation to under-utilisation frequently even without any traffic re-assignment. In such dynamic traffic situations, a precise method of moving traffic would prove to be more beneficial. That is why, we observe that when availability of network resource is low, the strict greedy approach outperforms the relaxed greedy approach.

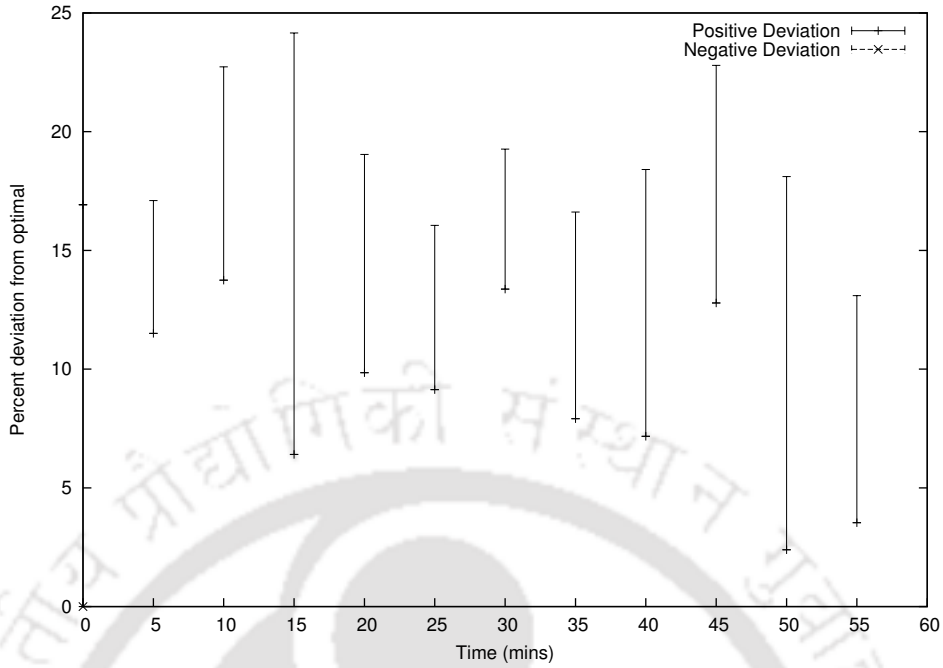
Our empirical results reveal that the strict greedy approach gives a consistent and relatively better performance in actual traffic scenarios where available bandwidth is dynamic. Hence, in the remaining part of our evaluation we pursue with the strict greedy approach only. The terms greedy approach and strict greedy approach are used interchangeably in the remaining part of our work.

5.5.4 Dynamic Network Environment

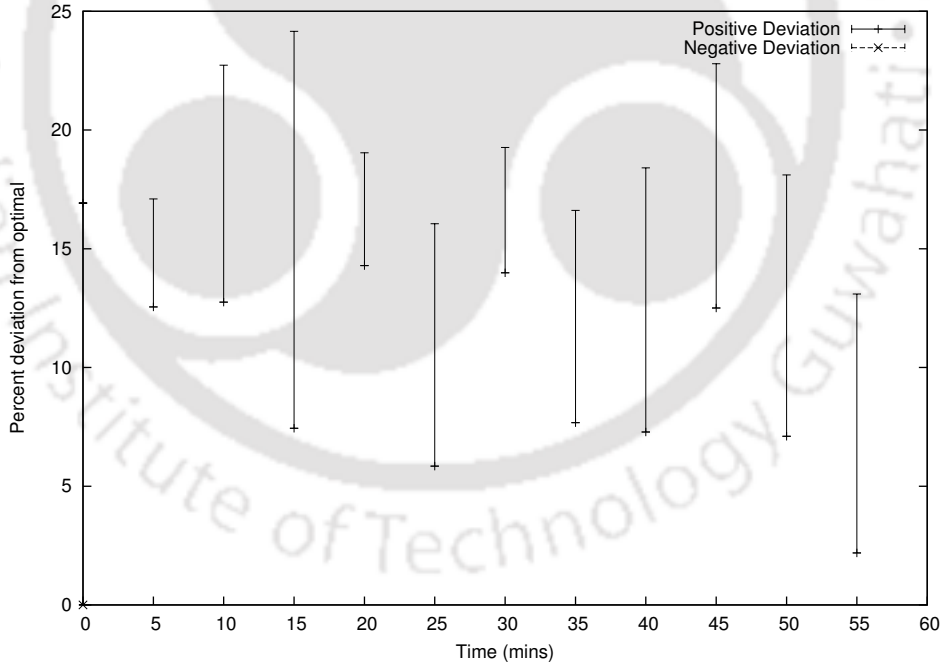
We have tested our route control mechanisms in a static network environment (no intradomain traffic present). Empirical results show that our approach results in good load balancing of the network traffic with a minimal re-assignment of the egress route of users. In this section we verify our route control strategy, proposed in section 5.4.3, that considers the dynamics of intradomain traffic too while selecting egress routes. The algorithm is verified using both synthetic as well as actual traffic traces on a wide range of network topologies. Finally we test the robustness of our approach to sudden drastic changes in the network load (level shift events).

Experiment Test-bed

In order to incorporate the dynamics of intradomain traffic into our simulation, we first need to create a topology of the internal network. The network topology, including delays and bandwidths and the sequence of user requests define a simulation. As far as choosing a network topology was concerned we used a standard topology generator (BRITE ([18]). The network model was initially created using the topology generator but later on enhanced by dynamically changing the link attributes, bandwidth and cost. To find how various attributes of network topology affect the load balancing of links we experimented with a set of network topologies. Our focus in these experiments was on access network topologies where users connect to access routers which in turn connect to edge routers and the Internet. Thus from the set of nodes we identified egress nodes and access nodes. To simulate the egress links between the stub network and its ISPs (*connecting path*), for each egress node we created a corresponding ISP node and created links between the two. During the course of the experiment, available bandwidth of the *connecting path* as well as the intradomain links were changed. Cost of a link is set to the inverse of its available

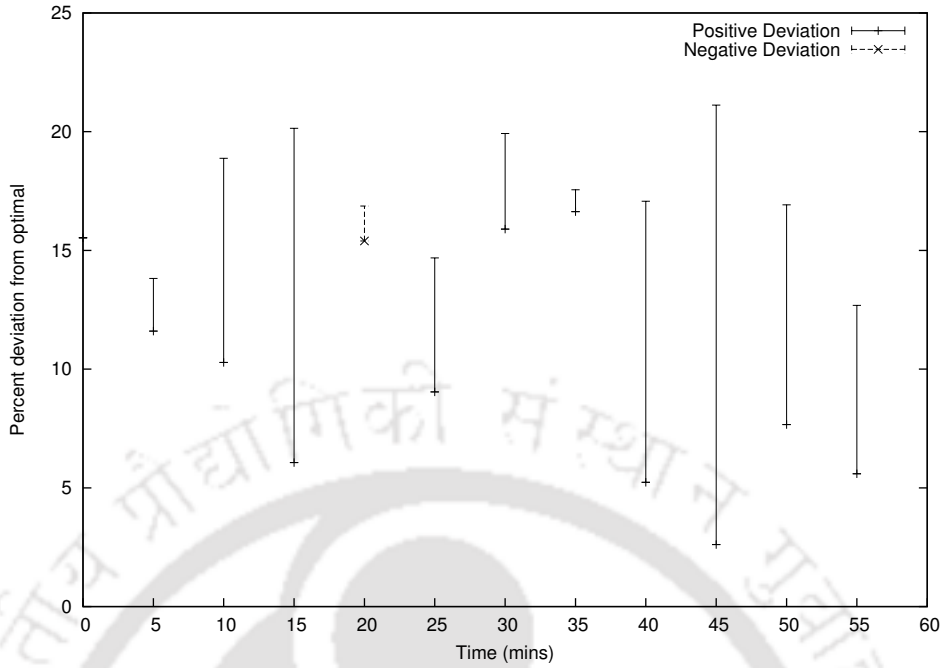


(a) Strict greedy

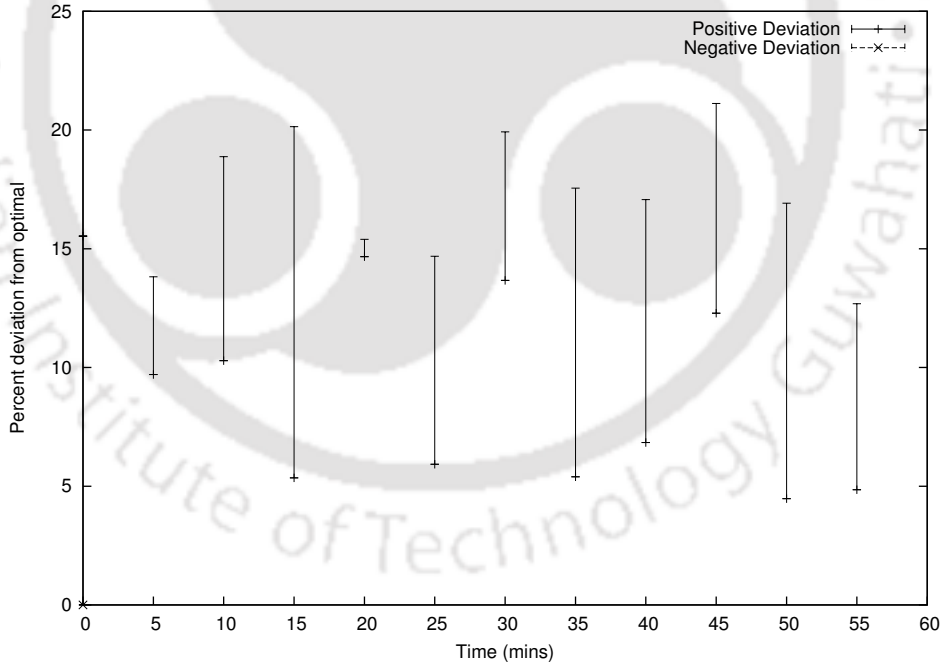


(b) Relax greedy

Fig. 5.10 Expt 3a. No cross traffic.

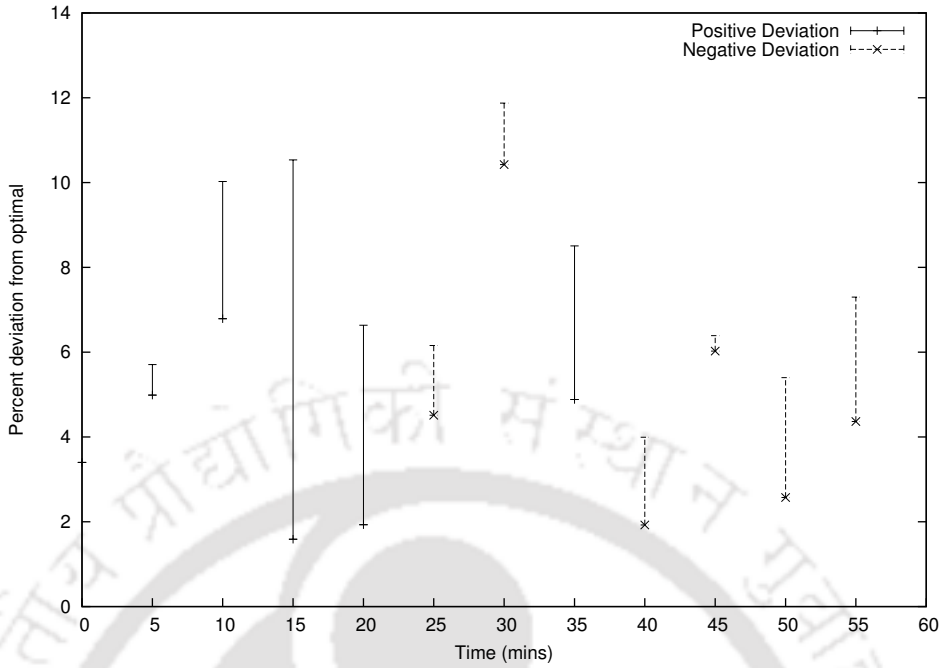


(a) Strict greedy

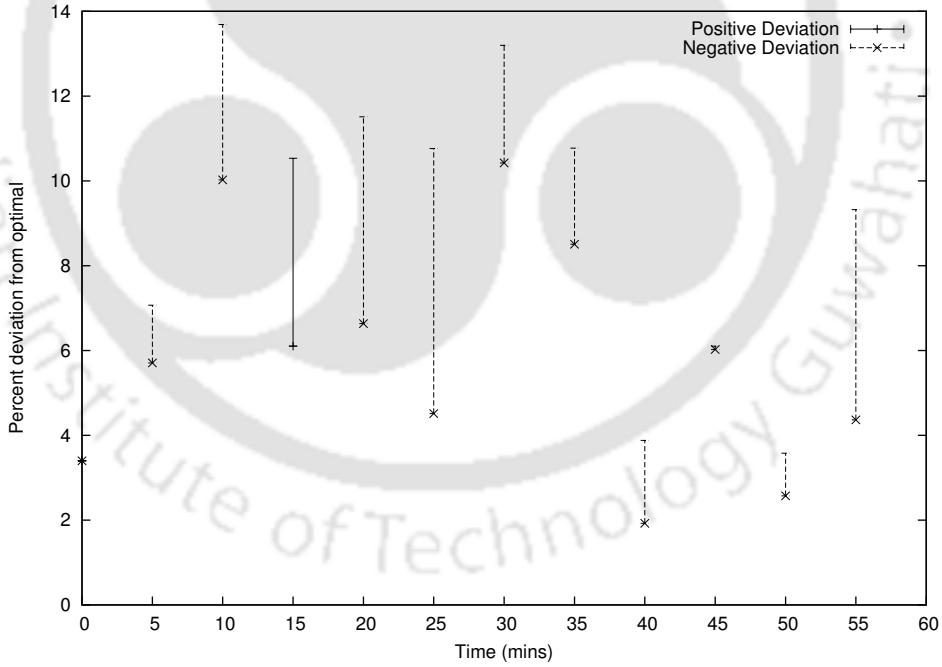


(b) Relax greedy

Fig. 5.11 Expt 3b. Moderate cross traffic.



(a) Strict greedy



(b) Relax greedy

Fig. 5.12 Expt 3c. Heavy cross traffic.

bandwidth. The intradomain protocol used was OSPF. The available bandwidth and as a consequence the link weights were computed at the end of each period. The simulation engine used was *ns-2* ([5]).

The users were attached to the access nodes. As the user nodes were created we concurrently create a destination node (sink) for each user and attached it to its corresponding ISP node. The cost of these two links was set to 1. With these experimental settings we simulated the *default* case, where no load balancing mechanisms were used. In the next run of our experiment we employed our proposed load balancing tactics. However, to simulate our route control techniques we dynamically need to change the egress route assignment of users. In order to effect user re-assignments, we created additional links from a destination node to each of the other ISP nodes. The costs of these additional links were initially set to a very high value (infinity). This means that traffic will flow from the first link only and the additional links will not be used. While simulating the greedy approach, if the heuristic required that a user be moved from its current ISP to a second ISP, the costs of the path from the destination node to the first ISP was changed to infinity and to that of the second ISP was changed to 1.

Figure 5.13 shows a network topology with two egress nodes and two ISPs. We created a user node *USER1* and attached it to an access node. We then created a corresponding destination node *DST1* for the user and attached it to *ISP1*. The cost of links between the destination node and the ISP and between the user and the access node were set to 1. OSPF was run within the network cloud. Client objects (for example a HTTP client) were attached to the user node and server objects (like for example a HTTP server) were attached to the destination node. During the simulation of the load balancing logic there would be a need to move the user *USER1* to the other ISP *ISP2*. To facilitate this we created an additional link between *USER1* and *ISP2*. The cost of this link was set to a

very high value (infinity). Hence traffic from *DST1* will be initially routed through *ISP1*. During simulation if the load balancing logic requires that *USER1* be re-assigned to *ISP2*, the cost of the link between *DST1* and *ISP1* will be set to infinity and the cost of the link between *DST1* and *ISP2* to 1.

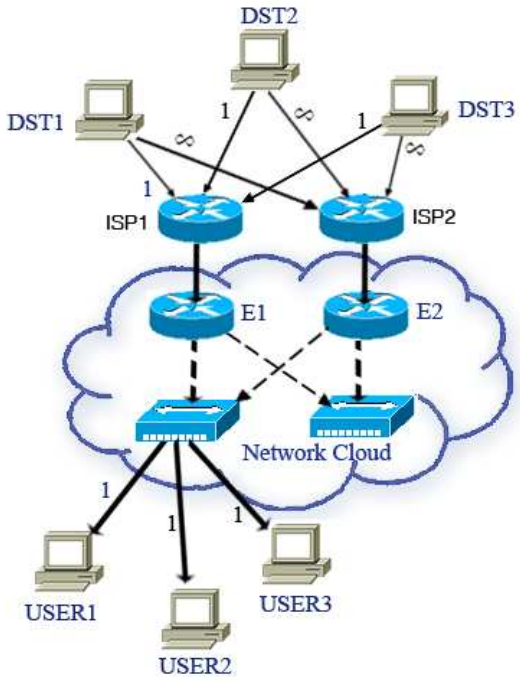


Fig. 5.13 Network topology for simulating synthetic data.

Experiment 4: Validating Using Synthetic Data

A major challenge while developing test beds for network experiments is modeling the Internet traffic. Our focus being on access networks, a primary user in our simulations has to be a consumer accessing the Internet to browse the Web and initiate downloads. The discrete event simulator *ns-2* provides a rich library of traffic models. In this simulation we used three different types of user classes represented by three different traffic models. Each instance of a traffic model represented an user type or user class in our experiment. The first user type modeled was by using the PackMime Internet traffic model ([35]), a model for generating HTTP traffic in *ns-2* simulations. *PackMime-HTTP* is the *ns-2* object to

model the interaction between clients and servers on a simulated link. The *ns-2* object consists of a client and a server cloud. The client clouds are attached to user nodes and the server clouds are attached to destination nodes. The number of HTTP connections between a client and server cloud was set to a value between 10 and 20. The second user type was simulated using the *ns-2* class *PagePool/WebTraf*, a standalone web traffic model ([49]). A PagePool object creates a session between a client and server object. The client object was attached to a user node and the server object was attached to a destination node. The number of sessions between a client and server varied from 10 to 100. Each session transferred between 100-1000 pages. Other parameters like interpage waiting time etc were generated using random functions. The third type of user class used in the simulation was a ftp server. A TCP connection was created between the destination and the user and a ftp server was attached to the TCP connection. The internal or intradomain traffic was generated using *Pareto On/Off* traffic sources.

In this experiment, the topology considered consisted of 25 nodes and 50 intradomain links. The number of egress nodes considered was four. Capacity of all the links were considered to be the same (10 Mbps). The duration of the experiment was 100 periods. To simulate the Internet traffic we created an instance of a user class and attached it to an access node. Instances of all the three user classes were attached to an access node. However, the number of users attached to each access node was different. Each user class also had a number of configurable parameters. The traffic generated by a user class to a large extent depends on the setting of these parameters. To bring about a disparity in the utilisation of the links, these parameters were set differently for different users of the same user class. The aim while manually assigning users to access nodes was not to balance traffic on the egress links. On the contrary we wanted to make the link utilisations highly imbalanced. In this experiment one of our objectives was to test the extent of performance improvement in load balancing possible on links with disparate utilisations. The other result that we

demonstrate is how load balancing traffic brings about improvement in round-trip time.

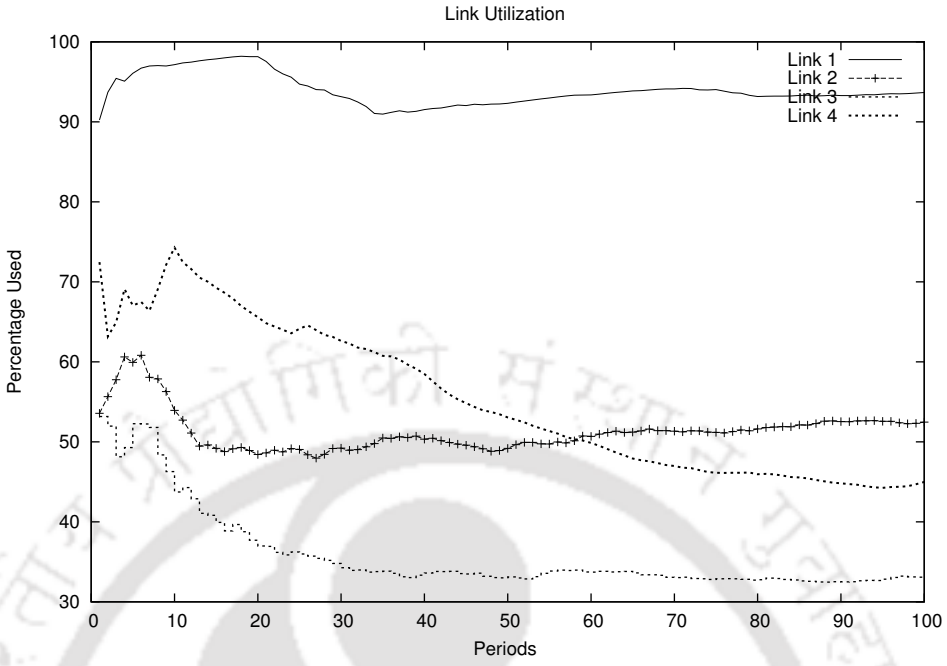
With the above experiment settings, we simulated the *default* case (no load balancing) in the first run of our experiment and collected the traffic traces. In the second run of our experiment we simulated the greedy heuristic with the same network and user class settings. The different user classes generate different amount of traffic. Even among users of the same class, traffic of a user depends on the user class settings. To make our experiment runs comparable we need to generate the same amount of traffic in each run of our experiment. A fundamental parameter for the first user type is the rate at which new TCP connections are initiated by a client. These as well as a number of other variables were stored for all users of this type during the first run of our experiment. In the case of the second user class, each client can generate a number of traffic sessions. The number of sessions as well as the pages per session, page size, inter-page arrival, object size etc was stored for all users that belonged to the second user class. For the third user type, we stored the number of ftp sessions. When the experiment was repeated to simulate the greedy heuristics, these stored parameters were used to generate the user traffic. These will ensure that the user traffic is similar in all the runs.

The link utilisations when no load balancing techniques were used is shown in figure 5.14(a). As can be seen there is a wide disparity in the utilisation of the links. The average deviation of the links from their ideal value (i.e. ranks) is 30.67 percent. The plot of the link utilisations when the greedy heuristic is used is given in figure 5.15(a). We find that application of our route control strategy significantly reduces the deviation of the link utilisations from their ideal value. The average deviation of the link utilisation from their ideal value is reduced to 6.03 percent. In order to underline the period-wise benefit of our route control strategy, we compared the deviations of the greedy approach from their ideal value for each period with the corresponding deviations of the *default* approach using a

stacked histogram, figure 5.16. In this figure, deviations for the first 25 periods are shown. A stacked histogram for each period allows comparison in the following way. The first stack depicts the least deviation; the next stack shows the next higher deviation and the amount by which it exceeds the previous stack. As can be seen, in the first and second period deviation of both the *default* and greedy approach is same. Hence, we can see only one histogram. In the third period the greedy approach has a lower deviation. As we can see from the plot, the deviations of the greedy approach are significantly lower than that of the *default* in all the periods. During our extensive experimentation we have observed that higher the traffic load imbalance for the *default* case, greater is the performance improvement of our load balancing schemes. In a real world scenario, traffic load of the *default* case will not be so highly imbalanced as was in the case of this experiment. Performance improvements are therefore likely to be less. The number of users re-assigned are shown in figure 5.17. The overall user re-assignment per period is about 4.48 percent. The number of users re-assigned is relatively high as compared to our previous experiment but at the same time improvement in load balancing has also been huge. This leads to the question, is the performance improvement in terms of traffic load balancing related to the number of users re-assigned? Our experiments so far indicate that the number of users re-assigned depends on the degree of traffic imbalance of the *default* case. However, in the later experiments we will find that this is always not the case. The performance improvement also depends on the network traffic scenario and heuristics used. We show situations where a higher user re-assignment actually leads to inferior performance.

In this experiment since synthetic data is used, we can store the request-response exchanges between a client and a server. In table 5.4, the outgoing traffic, incoming traffic and round-trip time experienced by the users is shown for both runs of the experiment. There is an improvement in round-trip time by about 7 percent as compared to the *default* case. The volume of outgoing and incoming traffic in both runs of the experiment is almost

the same.

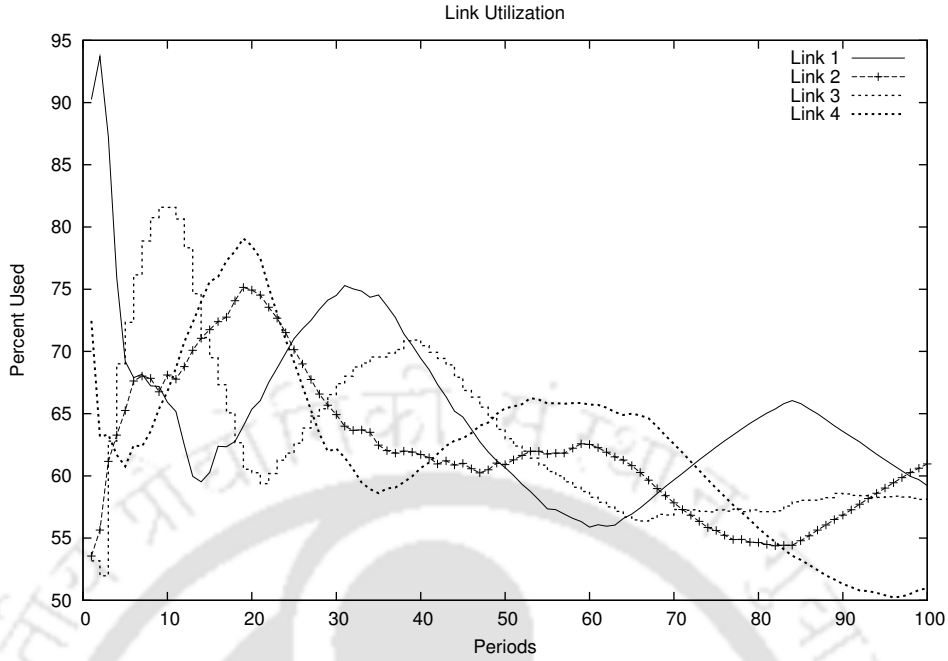


(a) Default

Fig. 5.14 Expt 4: Plot of link utilisations (contd.).

Table 5.4 Expt 4: Characteristics of user traffic.

Experiment	Outgoing Traffic (GB)	Incoming Traffic (GB)	RTT (seconds)
Default	0.17	2.47	990.88
Greedy	0.18	2.43	918.20



(a) Greedy

Fig. 5.15 Expt 4: Plot of link utilisations.

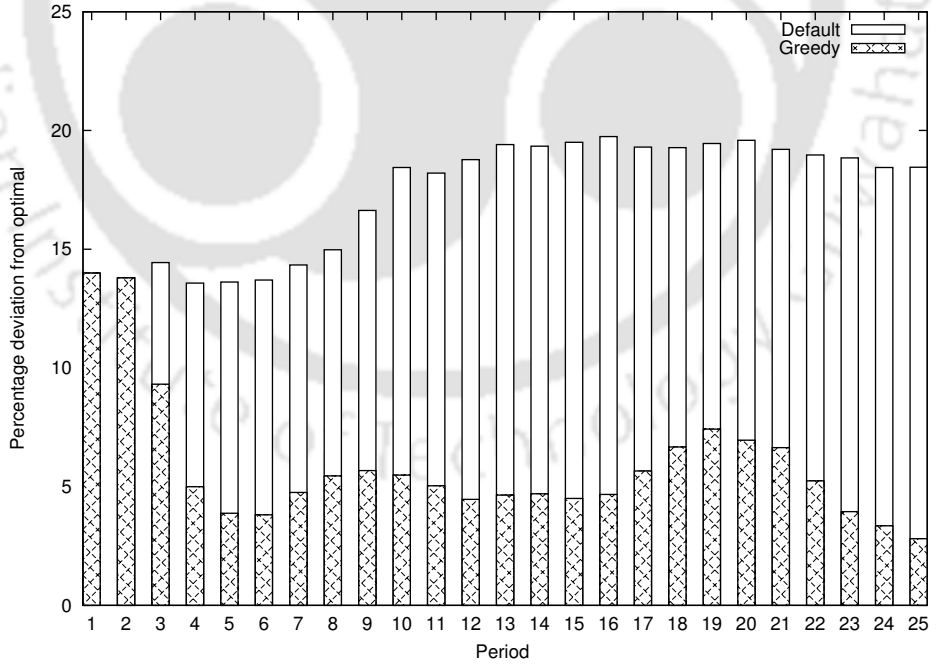


Fig. 5.16 Expt 4: Comparison of link utilisations using stacked histograms.

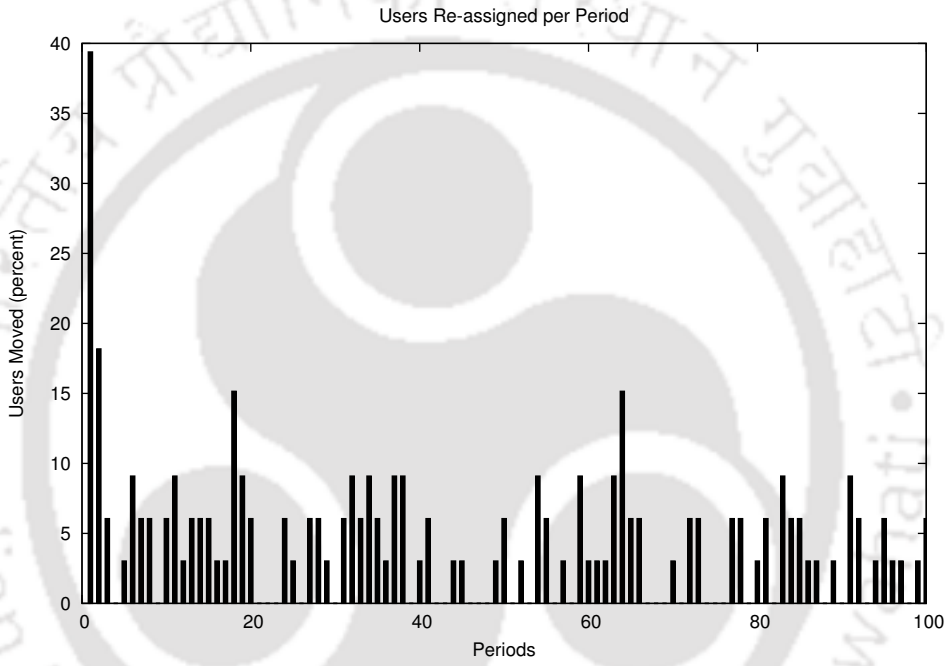


Fig. 5.17 Expt 4: Plot of users re-assigned

Experiment setup for Using Actual Traffic Traces

As Floyd and Paxson ([50]) has mentioned, simulating the Internet traffic is a difficult task due to the heterogeneous structure, immense size, and dynamics of the Internet. The Internet Engineering Task Force (IETF) too has realised that the offered traffic load as well as the underlying topology is crucial to measure the effectiveness of traffic engineering algorithms ([43]). With synthetic traffic, no matter which traffic model is used, one is not sure whether the essentials of actual Internet traffic have been preserved or not. Moreover, as we have seen traffic models have a number of configurable parameters and in order to make these models represent the wide range of Internet users, the parameters need to be configured individually. In the simplest case the behavior of a traffic model will depend on the distributions of the number of objects per page, the number of packets per object, the inter-arrival times of pages etc. If these parameters are chosen from the class of heavy-tailed distributions then the traffic will show a self-similar behavior. On the other hand if the parameters are chosen from an exponential distribution it will result in multi-fractal traffic behavior. The task of fine-tuning the user classes to represent the wide range of Internet users is therefore non-trivial. To thoroughly test our heuristics we need to expose our proposed algorithm to a wide range of traffic scenario. But developing models that exactly depict Internet traffic are open problems. Most of the traffic models proposed by researchers employ a certain level of abstractions to simplify the problem. Since the outcome of our simulation depends on the network traffic load, it is required that we use real, user-generated traffic to further substantiate our results. Thus the best possible way to validate our heuristics would be to use real traffic traces collected over a considerable period of time. As was done in the case of static network environment, we propose to use traffic traces collected from several egress links. Each of the egress links are represented in our topology by an egress node. To simulate the dynamics of Internet traffic we need to replay the traces on the network model but unfortunately *ns-2* does not support TCP connection from traces. The individual traces are, therefore, processed before being fed

into *ns-2*.

First from each of the traces we extracted the distinct users (IP addresses) present. We created individual ns2 trace files for each of these users. We scanned through a trace file and when an incoming packet for a user was encountered, the payload and inter-arrival time of the packet was appended to the ns2 trace file of the user. The inter-arrival time is the difference in time-stamp of the packet to the time-stamp of the packet last seen for the user. For each of the ns2 trace files we created a user node (USER1) and a destination node (DST1). The user node was connected to an access node and the destination to the related egress router (E1). The costs of these two links were set to 1. The trace was then streamed from the destination to the user. In order to simulate user re-assignments, we created additional links from the destination to each of the other egress nodes as was done in the last experiment. For example in figure 5.18, we created an additional link from DST1 to the second egress router (E2). The costs of such additional links were initially set to a very high value (infinity). While simulating the greedy approach, if the heuristic demands that the user be moved from egress link 1 to 2, then the cost from DST1 to E1 is set to infinity and cost to E2 is set to 1. The network model is shown in figure 5.18.

The cost of an intradomain link was set proportional to its capacity. The intradomain protocol used was OSPF. The other question was how to generate the internal traffic. A common practice is to run CBR (Constant Bit Rate) traffic between the backbone nodes. However, for the CBR traffic to have an effect on the route selection process, substantial amount of traffic needs to be generated. This in turn has an adverse effect on the running time of the experiments. In the last experiment we found that the intradomain traffic generated using CBR traffic did not have a significant influence in the selection of egress routes. Our endeavor in this experiment was to highlight the intradomain route changes induced by the incoming egress traffic. The intradomain cross traffic, therefore, should be substan-

tial. We assumed all intradomain links to have the same capacity and the internal traffic was distributed evenly across the whole domain. We simulated the influence of internal traffic by changing the capacity of the intradomain links every period. We experimented by dynamically changing the capacities of the intradomain links using different schemes - assign randomly generated link capacities, manually set the capacities to some pre-defined values or set the capacity equal to the average of the total ingress traffic. We observed a consistent and sufficient number of intradomain route changes were affected when the intradomain link capacities were set to a value equal to average of the total ingress traffic. The intradomain routes were re-computed at the end of every period to ensure that packets follow the latest intradomain paths.

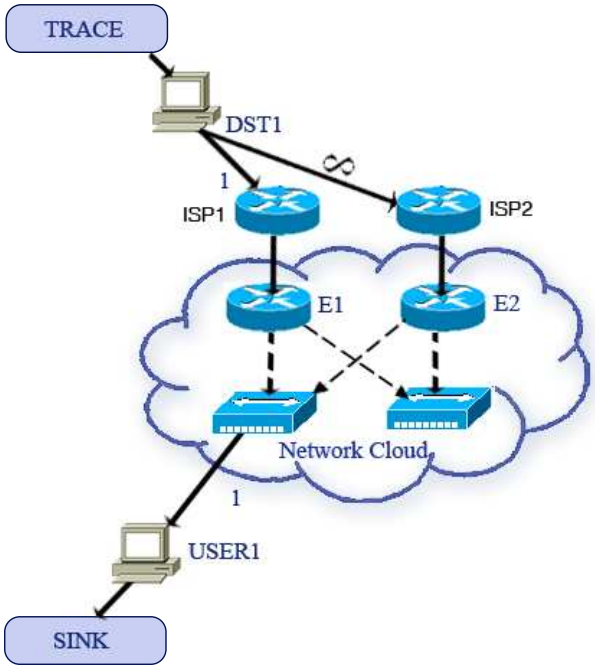


Fig. 5.18 Network topology for simulating actual traffic traces.

Experiment 5: Validating with Real Traffic Traces

We tested our dynamic route control technique with a wide range of topologies and actual traffic traces collected at different times of the day. In this section we show a typical result.

The user base considered was about 1500 users. The duration of the trace was 2 hours. The total traffic downloaded was about 20 gigabytes and the outgoing traffic was 1.36 gigabytes. The network topology consisted of 50 nodes and 300 intradomain links. The numbers of egress and access nodes considered were 8 and 17 respectively. With this experimental settings, we simulated the three cases - *default* case, the greedy algorithm that does not consider intradomain traffic and the greedy algorithm that considers intradomain traffic while selecting egress routes. The deviations of the links from their ideal value for all the three schemes were plotted using a stacked histogram. As can be seen in figure 5.19(a), in the first period all the three approaches have the same deviation and so we can see only one histogram. In the second period the greedy approach (without intra-domain traffic) has the least deviation which is followed by the greedy approach with intradomain traffic. The *default* case has the highest deviation in the second period. The average percentage deviation from the ideal value for the three approaches are - 14.51 (*default*), 8.64 (greedy, without considering intradomain traffic) and 9.02 (greedy, with considering intradomain traffic) respectively. The result of this experiment is similar to the results that we have seen so far in the previous experiments; the greedy approach performed better than the *default* case in all the time periods. However, when we consider a dynamic environment the performance improvement in terms of load balancing as compared to the *default* case is about 40 percent whereas we got more than 50 percent improvement in the static case. The user re-assignments are shown in figure 5.19(b). The overall percentage of users re-assigned is marginally higher than 2 percent.

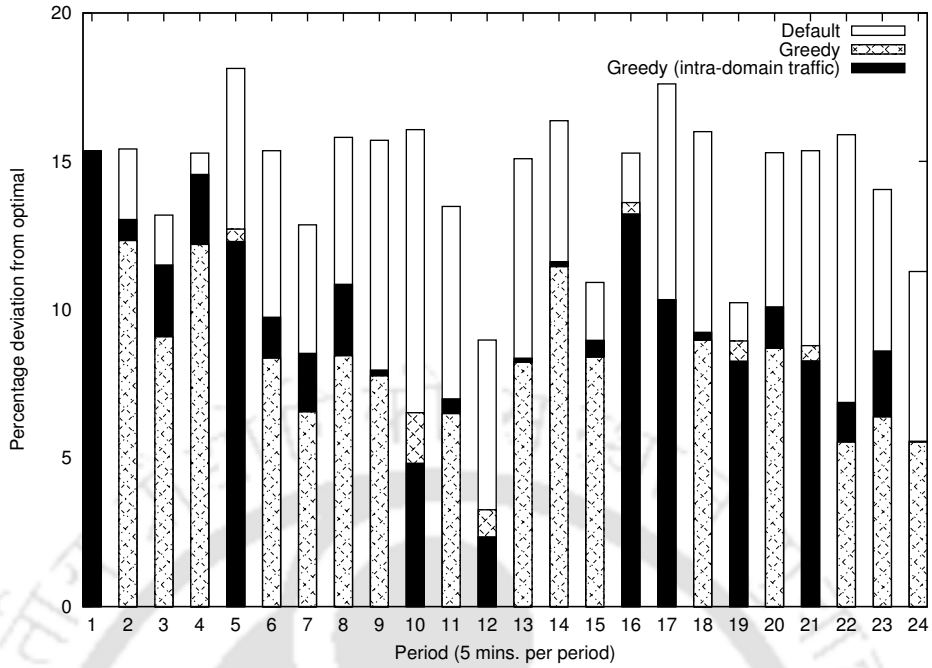
Frequency of Intradomain Route Changes

In an actual network, latency of an incoming traffic flow will depend on the delay incurred by the flow on the Internet path, delay on the intradomain paths as well as delay incurred by the flow's corresponding outgoing requests/acknowledgments. This means for different egress routes chosen for a user, the latencies incurred by the incoming traffic of the

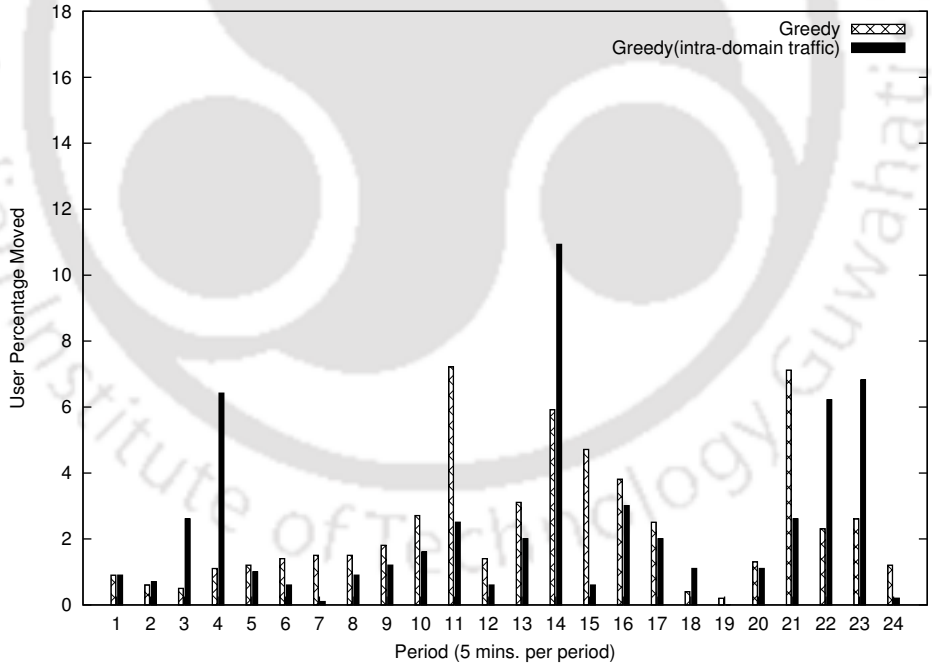
user will also differ. However, in our simulation we cannot see the effect of such delays as the incoming traffic is pumped at a pre-defined rate. Nevertheless, to bring to light the influence of intradomain traffic, we recorded the intradomain routes followed by the greedy approaches without and with considering intradomain traffic. At the end of each period we compared the intradomain paths followed by a user for both the heuristics. The frequency of intradomain route changes in each period is tabulated below (table 5.5). For this experiment, we found that on an average the difference in path followed by the two greedy heuristics is about 14 percent. In other words, when we consider the intradomain cost into our heuristics, packets will follow a lower latency path about 14 percent of the time. To substantiate our results further, during our simulation of the greedy approaches whenever a user was re-assigned we output the intradomain path followed by the user as well as the IGP cost of the path. The total cost of intradomain paths followed by users when egress links are selected based on the link rank is 1295.91. In contrast if the egress links are selected based on their IGP cost, the total cost of the intradomain paths is 590.65. Thus there is an overall improvement by more than 50 percent in terms of the IGP cost.

Table 5.5 Expt 5: Frequency of Intradomain Route Changes.

Period	Frequency
1	0
2	6.77
3	12.5
4	18.75
5	13.54
6	17.19
7	11.46
8	17.19
9	12.5
10	16.67
11	14.58
12	19.27
...	



(a) Comparison of Ranks



(b) Percentage of Users Re-assigned

Fig. 5.19 Expt 5: Greedy approach without and with intradomain traffic.

Experiment 6: Examine Robustness to Level-Shift Events

It is well-known that traffic variability of link loads will differ at different time scales. The smaller the time period considered the higher will be the variability and burstiness of the trace. In terms of network traffic characterisation, a scale of 5 minutes (the time period considered in our simulation) is regarded to be quite coarse. Although the traffic characteristic can be chaotic at such time scales, measurement analysis has shown that variability of the incoming bytes and packets will not be drastic between periods ([39]). In our proposed solution, we provision network resources for the next time period based on the current measurements. The implicit assumption made in our solution is that available bandwidth of links will not drastically vary between successive periods. In other words we assume that the utilisation of links will gradually change. This assumption will hold in normal scenarios, but there may be abrupt changes in the available bandwidth. Sudden drastic changes in the available bandwidth are regarded as level shift events. Such level shifts can occur due to denial of service attacks, link failures, route flaps or other network anomalies. In this experiment we want to test the responsiveness of our proposed route control technique to level shifts and intense traffic spikes. Researchers model level shift events by re-playing the same traffic trace more than once ([71]). On the other hand denial of service attacks are modeled using probability distributions like Monte-Carlo sampling. While which of these models best describes a level shift event is debatable, the net effect is that all of these models attempt to exhausts the network resources. The other important point that needs to be considered while simulating level shifts is that such events occur for relatively short durations. In this work we propose to model level shifts by reducing the available bandwidth of a few links by half in a period.

We repeated the previous experiment (experiment 5) with the same input traffic and available bandwidth. To create the level shifts, from out of the 8 egress links we reduced the bandwidth of two links, link 4 and link 5, by half. The bandwidth reduction was affected

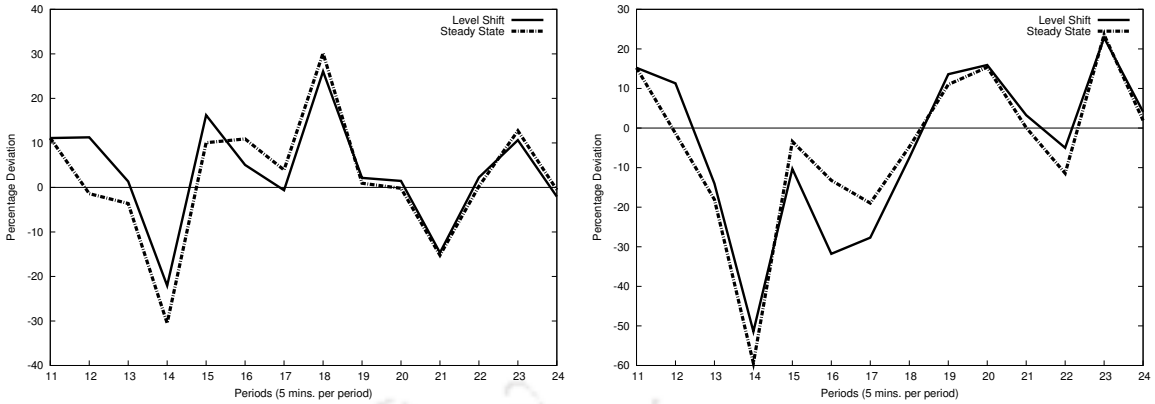
in the middle of the simulation and for a brief duration (periods 12 to 15). However, for any route control scheme to effectively distribute the network traffic, the total available bandwidth must be more than the total traffic entering the network. In this experiment we, therefore, ensured that the total traffic entering the network was less than the total bandwidth available. The average deviations of the links will obviously deteriorate due to a level shift event for the greedy approach. The average deviation of link utilisation from its ideal value for the greedy approach now is 13.12 as compared to 9.02 before. But the points that we want to investigate are how our route control scheme spreads out the effect of the sudden reduction in bandwidth to the other links and how long does it take for the links to converge to their steady state value after a level shift event. The deviations measured when no level shifts occur were considered as the steady state values. Thus, we compared the deviations observed in this experiment with those values recorded in the previous experiment. In figure 5.20 and 5.21 we plot the deviations of all the eight egress links and compare with the corresponding values of the previous experiment (their steady state values). The positive deviations indicate the link is under-utilised and negative deviations denote that the link is over-utilised. As expected up to the eleventh period, deviations of all the links were exactly same as with those of their steady state values. From the twelfth period on wards the deviation of all the links increased. We found that a link that was previously over-utilised had become under-utilised and vice-versa. While the deviation for links 4 and 5, whose available bandwidth had been drastically reduced was effected the most, in general the deviations of all the others links also showed an increasing trend. Due to the way we compute the ideal value of the links, our route control approach had been successful in distributing the effect of the level shift. Disturbances in the link deviations were visible even after the level shift event was over, that is after the fifteenth period. But from the eighteenth periods on wards, plots for all the links coincided with their steady state value. In this experiment, we found that our algorithm took at most three periods to converge to its steady after a level shift event had occurred. The number of user re-

assignments also show an increasing trend (figure 5.22) when level shifts occur. This is expected because more traffic needs to be realigned so as to neutralise the effect of the level shift event.

5.6 Conclusion

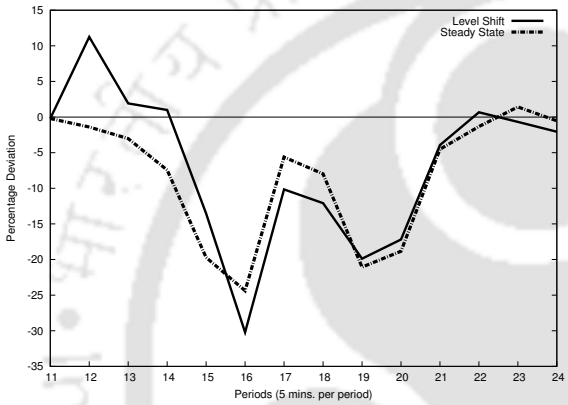
In the previous chapter we have shown practically how load balancing traffic on the egress links of an end network results in overall round-trip time improvements. In general, latency experienced by users of a link will increase as utilisation of the link increases. Load balancing the traffic will ensure that variations in latencies are reduced. In this chapter we proposed to re-distribute the traffic of a multihomed end network in proportion to the available bandwidth of the links. This value is referred to as the ideal utilisation of a link. We showed that the problem of load balancing traffic is NP-complete even when unrealistic assumptions about the input traffic were made. Leveraging upon our detailed analysis of the problem we proposed heuristics. The heuristics iterate recursively in periods using the measured parameters of the previous period to determine the course of actions in the next period. To measure the improvement in load balancing, we compared the results of our route control scheme to that of the *default* approach, where no load balancing schemes were used. The experiments were performed using actual traffic traces collected from the transit point of our institute as well as synthetic data. We found that the deviation in link utilisation for the *default* approach from our so called ideal value ranged from 15 - 30 percent. Employing our route control schemes reduced these values by about 40 percent. Further we showed that the improvement in performance is uniform throughout the duration of the simulations. The load balancing of the links were achieved by re-assigning about 2 percent of the users.

In this chapter we basically proposed two load balancing techniques - greedy algorithm and random user selection (RUS) algorithm. Both the algorithms behave similarly, except

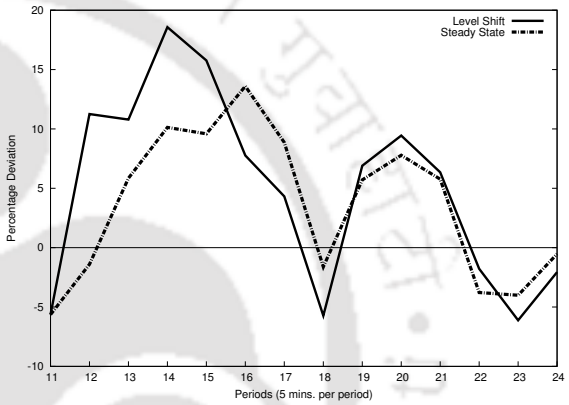


(a) Link L0

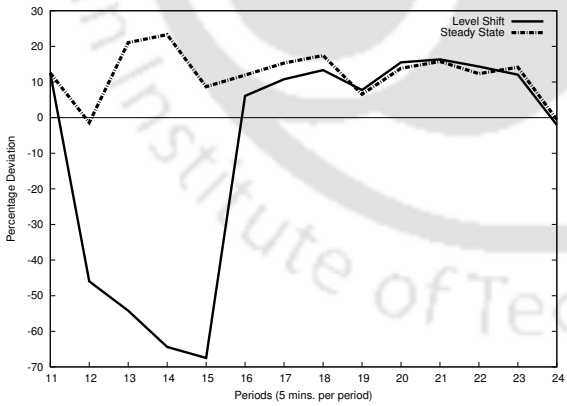
(b) Link L1



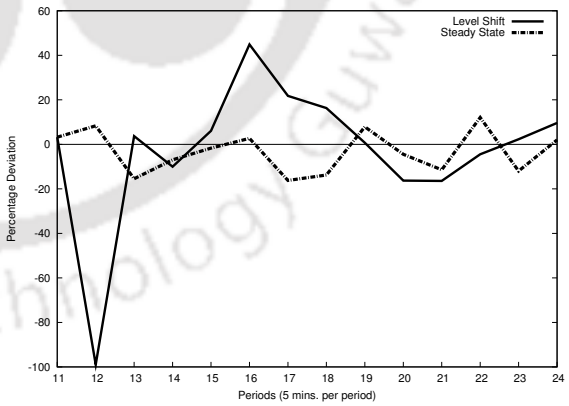
(c) Link L2



(d) Link L3

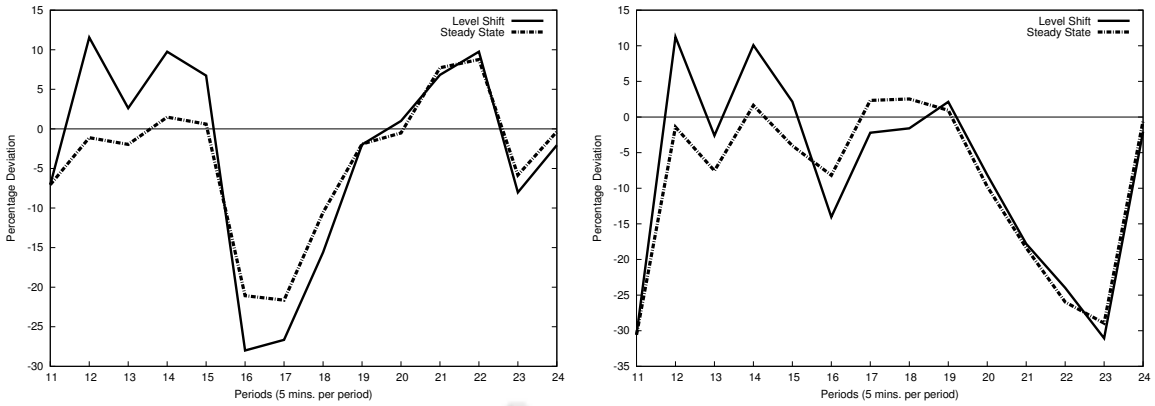


(e) Link L4



(f) Link L5

Fig. 5.20 Expt 6: Comparison of link deviations with their steady state values during a level shift event (contd.).



(a) Link L6 (b) Link L7

Fig. 5.21 Expt 6: Comparison of link deviations with their steady state values during a level shift event.

that in the former we sort the users before considering them for re-assignment whereas in the later users are picked up randomly for re-assignment. The performance of both these techniques was comparable in terms of load balancing the links. In terms of user re-assignment, our experimental results showed that RUS is costlier by about 1 percent. On the other hand if we compare the computational complexities, RUS has a lower run-time of $O(S)$ as compared to the $O(SlogS)$ of the greedy approach, where S is the total number of users of the domain. Given the enormous computational power available today, user re-assignments are likely to be costlier than the computational power required. Therefore, the greedy approach is recommended. Moreover, in the greedy heuristic we do not need to sort all the users but only those users assigned to an over-utilised link. In practice the number of users that require to be sorted will be much less than S. To prevent users from oscillating between links, we proposed that once a user was re-assigned it should not be considered for re-assignment for a certain time interval. In our experiments we restrained a user from being re-assigned twice, for about 2 hours without affecting performance. This restriction, other then preventing user oscillations also has the advantage that it allows the BGP route updates, that are affected every time a user is re-assigned, to converge.

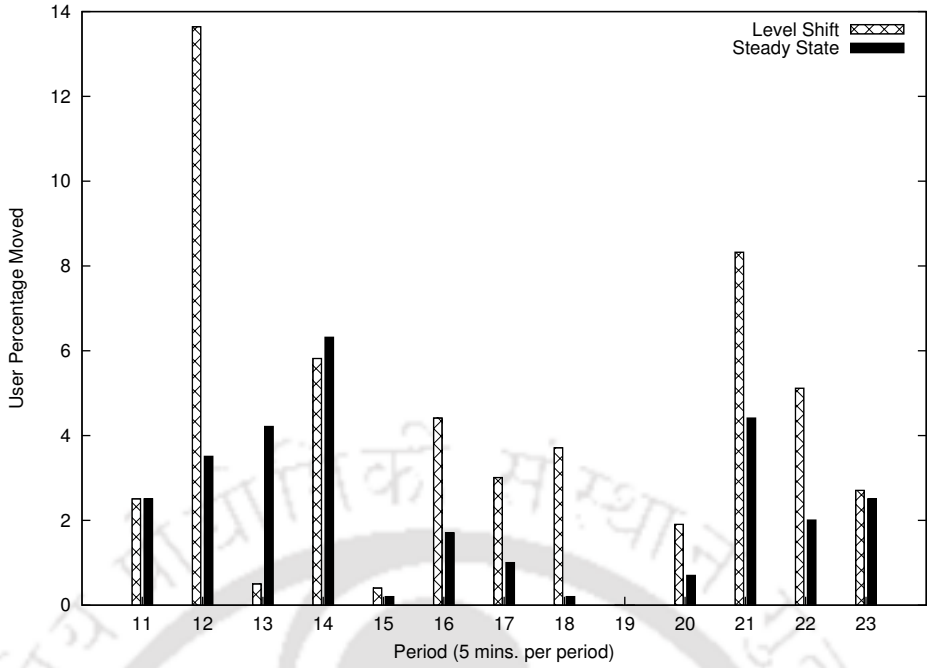


Fig. 5.22 Expt 6: Comparison of user re-assignments during a level shift event.

The greedy approach checks whether a user *fits* a link before actually re-assigning the user. We found that in certain traffic conditions, especially when the number of users present is limited, skipping this test does result in better performance (relaxed greedy approach). However, the relaxed greedy approach does not perform well in dynamic traffic conditions. On the other hand, the greedy approach gave a consistent performance in all traffic scenarios. We found that when intradomain traffic dynamics was considered during the egress route selection process, the performance was relatively lower. However, traffic within the network followed a lower cost intradomain path. Thus the RTTs are expected to improve. Finally we showed that our route control strategy can spread out the effect of sudden reduction in bandwidths. Experimental results showed that our route control strategy converged quickly after a level shift event.

Chapter 6

Distributed Route Control Strategies

6.1 Introduction

The Internet architecture was first proposed in the early 1970s and since then its design philosophy has evolved considerably and it is still evolving. The architecture is flexible and it allows new services to be added which is one of the reasons for the growing popularity of the Internet. Nevertheless, the architecture has certain fundamental goals that need to be preserved ([40]), like for instance the ability to provide communication service in the face of failing networks and gateways. One such goal of the Internet architecture is to provide distributed management of its resources. Keeping up with the spirit of the Internet, in this chapter we strive to present a distributed version of our route control technique.

Centralised routing protocols have the advantage that we need to advertise less information, administration is easy and algorithms are simple. In a centralised architecture we assume that there is a manager or router where all the algorithms run. The manager uses some sort of signaling mechanism to see the current state of the network. Equipped with the latest network information, the manager allocates network resources and implements route control strategies. The manager is also responsible for setting up of the routes from the user to the egress routers. In such centralised architectures, the reliability of all routing

processes is highly dependent on the manager. The manager, therefore, becomes a major source of bottleneck as the network size scales. Moreover, centralised routing approaches are also not consistent with the design philosophy of the Internet. Further, due to geographical distances, collecting all the required information at a central router may be costly.

In a distributed scheme, we assume that the same algorithm runs in a distributed fashion on all nodes of the network. The nodes independently make routing decisions on the basis of network information received by them. One of the principal advantages to distributed routing is its scalability. However, it requires that the current network status be disseminated to all the nodes. In practice we find many standard Internet algorithms where the network information needs to be broadcast to all nodes of the network. Path selection algorithms like *QoS routing* rely on current link state metrics to select a path that meet the QoS requirement. Information about availability of resources is used by *call admissions* algorithms to reject a new flow when the cost of the path is deemed too high. In response to the requirement for current network status by a large number of traffic engineering architectures, a framework has been proposed to distribute link state information within a domain. OSPF-TE ([24]), this extension to the OSPF protocol allows distribution of additional link state information along with the link state updates - available bandwidth, propagation delay and hop count. In this work we assume that each border router maintains an up-to-date database of the network topology which includes available bandwidth and utilisation of the egress routes. These two metrics are presumed to be included in the link state advertisements (LSAs) broadcast by the border routers.

A fundamental goal in a large scale de-centralised system is how to best utilise the available resources so as to optimise the overall system performance. The objectives of our distributed scheme are the same as that of the centralised one which has already been well-defined and analysed in the previous chapters. Our approach, therefore, is to

map the schemes proposed for a centralised architecture to a distributed environment. We do an in-depth analysis of our centralised approach and identify those functions that can be re-used in a distributed system. The challenges in making our centralised route control schemes distributed are highlighted in section 6.3. In section 6.4, we theoretically show using probability analysis that if users are re-assigned using a certain probability distribution, the expected performance of the distributed scheme will be comparable to that of the centralised one. We then present distributed heuristics for both a static as well as a dynamic network environment. Section 6.5 present the results of our simulation. We validate our proposed distributed heuristics by comparing the results with those of the corresponding centralised heuristic.

6.2 Assumption

The assumptions about network traffic, topology and user behavior are the same as those we had made for the centralised approach. We reiterate below some of the assumptions that are fundamental for the distributed approach.

i. The intradomain protocol deployed has traffic engineering extensions. Border routers include utilisation and available bandwidth of their egress links in the periodic updates broadcast.

ii. Each user or node is aware of its current egress route assignment.

iii. A user can measure the traffic it receives in a period. Continuing with our earlier convention, it means a user i knows $b_i(t)$, the amount of incoming traffic it received in the time period t-1 to t.

The second and third assumptions were implicit in the centralised approach. Since the manager was responsible for assigning egress routes to users it was implicitly assumed that the manager knew the route assignments of all the users. The manager was also aware of the current network traffic status hence it could ascertain the traffic received by each user.

6.3 The Distributed Problem

In this section, we analyse our centralised route control scheme, in particular the greedy heuristic for static networks proposed in chapter 5 (section 5.4.2). We highlight the difficulties of deploying the centralised heuristic in a distributed environment. We examine and identify those modules that can be re-used in a distributed approach and those that require modification. A summary of the steps of the greedy approach is given in table 6.1.

Table 6.1 Greedy Approach: Summary of Modules

Module 1: Implemented using Algorithm 1
i. Compute total available bandwidth, total incoming traffic and utilisation of the individual links. ii. With the metrics obtained above compute the ideal utilisation of each link. The product of total incoming traffic and available bandwidth of a link divided by the total available bandwidth is the ideal utilisation of a link. iii. The difference of ideal utilisation and utilisation gives measure of the rank of a link. iv. Enumerate the over-utilised and under-utilised links. Over-utilised links will have a negative rank and under-utilised links will have a positive rank.
Module 2: Implemented using Algorithm 2
i. Select all users assigned to over-utilised links and sort them in descending order based on the input traffic received by them. ii. Select the first user from the sorted list. iii. Get an under-utilised link. Check if the user *fits* the link, if so re-assign the egress route of the user. iv. Finally update utilisation and rank of both the links.
Module 3: Implemented using *getnext* function
Scan through the under-utilised links and return the most under-utilised link.

In the centralised approach, the first task was to compute ranks of the links. Since we assume that utilisation and available bandwidth metric of all the links are flooded throughout the network, each node can compute ranks of the links in a distributed fashion using the same procedure, algorithm 1. Moreover, each user knows its current egress route assignment. Thus a user can trivially determine whether it is assigned to an over-utilised link. The users can also identify all the under-utilised links.

In a distributed approach, the algorithm will be run individually by each of the nodes which effectively mean that the total number of users seen by the algorithm is only one; hence the need to sort users does not arise. The next step of the greedy approach is to select from the list of users assigned to over-utilised links, one user at a time and check if their egress routes need to be re-assigned. In the distributed approach, users will decide in parallel whether to re-assign themselves or not, thus considering the users one at a time is not possible unless there is some coordination amongst them. Further after each user re-assignment the greedy approach updates the status of the links. The decision of whether to re-assign a user or not depends on the current state of links. In a distributed approach, each node will maintain the status of the links locally and it will not be aware of the decisions taken by other nodes. The requirement that as users get re-assigned the status of the links also need to be updated globally and concurrently is the main impediment in making our centralised heuristics distributed.

Again during the process of selecting an under-utilised link, in the centralised approach the selection was based on the current state of the links. The most under-utilised link was selected. As users get assigned to the most under-utilised its utilisation will change and after a point it will no longer remain the most under-utilised link. As a result another link will get selected. In this way the excess traffic load of the over-utilised links gets proportionately distributed among the under-utilised links. For this scheme to work in a distributed environment again the users need to have a global view of the link status and the ability to update the status concurrently as users get re-assigned. Otherwise, it will result in all users selecting only one link, the most under-utilised link.

To conclude, the two main challenges in making our centralised heuristic distributed are: (i) in the centralised approach, the process of assignment of users was serialised whereas in the distributed approach it is done in parallel. (ii) the status of the links should be

maintained and updated globally as users get re-assigned. From an implementation view point it means we can re-use module 1, while module 2 and 3 will require modifications.

6.4 Distributed Load Balancing Techniques

In this section, we view our load balancing route control technique from a different perspective. One would have observed that the principal idea of our load balancing algorithm is to move out traffic from an over-utilised link such that its utilisation becomes ideal. The rank of an over-utilised link is a measure by which utilisation of the link exceeds its ideal value. The centralised heuristic, therefore, attempts to move out traffic equal to its rank from an over-utilised link. In the distributed approach if we allow all users assigned to an over-utilised link to decide independently whether to re-assign its egress route or not such that the net traffic moved out from the link matches that of the centralised heuristic then our goal will be achieved.

Suppose there are L users assigned to an over-utilised link. The total number of assignments possible is L^2, including the cases where no users get re-assigned or all users are re-assigned (an unlikely scenario). From among these possible assignment plans we have to select one that best meets our requirement or the one pursued by the centralised approach. Since we assume there is no coordination among the nodes, a deterministic approach will not be possible. We need to select one of the assignments randomly. Decision of the users therefore has to be based on probability theory. We claim that in the distributed approach, if all users assigned to an over-utilised link re-assign themselves with a constant probability (P_c), equal to the ratio of the rank of the link to its utilisation, then the expected amount of traffic moved out from the link will be equal to its rank. A formal proof of this claim is given in theorem 3.

Theorem 3. *Let o be an over-utilised link with rank rank[o] and utilisation U_o. If all users*

assigned to the over-utilised link re-assign themselves with a constant probability P_c, where $P_c = \frac{rank[o]}{U_o}$, then the expected amount of traffic moved will be equal to the rank of the link.

Proof. Let X be a random variable that has a value 1 if a user is moved and 0 if not moved. We define the probability mass function of X as

$$Pr(x) = \begin{cases} P_c, & \text{if } x = 1 \\ 1 - P_c, & \text{if } x = 0 \end{cases}$$

Clearly X is a Bernoulli random variable. By definition the expected value of a Bernoulli random variable X is $\mathbf{E}(X) = P_c$. Let L be the total number of users assigned to the over-utilised link o. Then expected number of users moved is $\sum_{i=1}^{L} P_c$. If b_i is the incoming traffic of a user i, then expected amount of traffic moved is:

$$\sum_{i=1}^{L} P_c b_i = P_c \sum_{i=1}^{L} b_i = \frac{rank[o]}{U_o} U_o = rank[o]$$

□

6.4.1 Static Environment

Theorem 3 suggests that in the distributed route control scheme if users are moved with a pre-defined constant probability, the expected amount of traffic moved from an over-utilised link will be comparable to that of the centralised scheme. The next issue is how we select an under-utilised link. This will depend on the type of network environment we consider. In the case of a static network environment, where input traffic is assumed to be unknown and available bandwidth can vary but intradomain traffic is steady, the technique adopted was to select the most under-utilised link. We have already indicated such an approach will not work for the distributed scheme since utilisation of the links is not updated concurrently with user re-assignments. Moreover, as the users are not sorted and assigned serially, selecting the most under-utilised link does not make sense. Ideally

the quantity of traffic that should be additionally assigned to an under-utilised should be equal to its rank. In the distributed approach we therefore propose that a user selects an under-utilised link with a probability equal to the rank of the link. The expected amount of users assigned to each under-utilised link will be proportional to their rank. Putting together the pieces, a comprehensive summary of the distributed heuristic is given below:

i. Compute rank of the links based on the current network traffic state using algorithm 1. Enumerate the over-utilised and under-utilised links.

ii. A user checks if it is assigned to an over-utilised link. If *yes* then go to the next step else *exit*.

iii. Based on theorem 3, the user plays a random game and decides whether it will re-assign itself. If *yes* then go to the next step else *exit*.

iv. From the set of under-utilised links select one based on the rank of the link.

Step one and two can be realised using our earlier algorithm 1. A pseudo-code of the third step is given in algorithm 6. The decision of re-assigning the egress route of a user is based on the outcome of a random game. The outcome of the random game in turn depends on our definition of P_c. Using our definition of a link rank, the value of the probability P_c for an over-utilised link o can be re-written as shown below, where IU_o is the ideal utilisation of the link:

$$P_c = \frac{\|(IU_o - U_o)\|}{U_o}$$

The value of P_c will be less than 1. The random game to decide whether a user will be re-assigned or not is designed as follows. Generate a uniformly distributed random number between 0 and 1. Compare the random number with P_c. If the generated random number is less than P_c, re-assign the user else exit.

The *getnext* function implements the procedure to select an under-utilised link based on the probability of the link's rank. Again which under-utilised link will be selected is based on the output of a random number generator. However, irrespective of the output of the random number, every time the function is invoked it should return an under-utilised link if one exists. We sum the rank of all the links and compute the relative range each link can take. Suppose there are three under-utilised links with ranks r_1, r_2 and r_3. Let $S = r_1 + r_2 + r_3$. The relative percentage of the link ranks will be: $p_1 = (r_1/S) * 100$, $p_2 = (r_2/S) * 100$ and $p_3 = (r_3/S) * 100$. The relative range of the first, second and third links can be respectively - 1 to p_1; $(p_1 + 1)$ to $(p_1 + p_2)$; and $(p_1 + p_2 + 1)$ to 100. We generate a uniformly distributed integer random number 1 and 100. Depending on which range the random number lies, the function returns the corresponding link.

As in the centralised case, before a user actually re-assigns itself it may check whether it *fits* the under-utilised link or not. In a *strict* implementation of the distributed approach only users that *fit* an under-utilised link are allowed to move whereas in a *relaxed* implementation no such checks are made. However, in this case since the link utilisations are not updated concurrently along with the user assignments, checking this is not possible, and the implementation will have to be *relaxed*. The run time of the distributed heuristic is $\mathsf{O}(N)$, where N is the total number of links. The centralised greedy approach had a higher run-time of $\mathsf{O}(SlogS)$ as it required sorting of the users where S is the total number of users. In the distributed approach there is no scope for sorting of the users. The distributed approach is, therefore, no longer greedy. So what about the number of users re-assigned? Since users are selected for re-assignment based on their *relative over-utilisation*, the distributed approach is in a way similar to the centralised Random User Selection (RUS) approach proposed in chapter 5 (section 5.4.2). The expected number of users re-assigned will be the product, $L.P_c$ where L is the number of users in the over-utilised link. The distributed algorithm being based on a probability analysis, simulation

results show that the actual traffic re-assigned is less than the deterministic approach of the centralised scheme. Consequently the number of users re-assigned is also relatively less.

Nevertheless, could we have designed our distributed approach to emulate the behavior of the greedy approach? That is the probability of a user being re-assigned depends on the volume of its incoming traffic. This is possible if the value of P_c was set as given below:

$$P_c = \frac{b_i.rank[o]}{\sum_{i=1}^{L}(b_i)^2}$$

With such a value of P_c, the probability of a user being re-assigned would be proportional to its incoming traffic and yet the amount of traffic re-assigned will be equal to the rank of the link. However, this scheme will require that all the $b_i's$ are included in the periodic updates. In a network there will be potentially a large number of users, such a proposition is not practical.

6.4.2 Dynamic Environment

In actual network conditions not only does traffic and available bandwidth vary but intradomain traffic changes dynamically too. As per our prescribed design objectives, a full-fledged distributed route control architecture must also consider the intradomain traffic dynamics while allocating egress route to users. The scheme that was followed in the centralised architecture was to select from the set of under-utilised links the one with the least IGP cost. In a distributed environment, the performance of this scheme will depend on how users are physically distributed across the network. If users who want to re-allocate themselves are well spread out across the domain, the cost of reaching the under-utilised links will differ for the different users. As a result the excess traffic load will reasonably get disseminated amongst the under-utilised links.

In general our observation is that users assigned to an over-utilised link are attached

input : A[1..N], U[1..N]

/* A[]: Link Available Bandwidth */
/* U[]: Link Utilisation */
/* x[i]=e, user i assigned link e */

for $e = 1$ **to** N **do**
  $T\_Abw \leftarrow T\_Abw + A[e]$
  $T\_traffic \leftarrow T\_traffic + U[e]$
end

/* Get the list of under-utilised links */
for $e = 1$ **to** N **do**
  $IU[e] \leftarrow \frac{T\_traffic \,.\, A[e]}{T\_Abw}$; /* Ideal Utilisation */
  $rank[e] \leftarrow IU[e] - U[e]$
  if *rank[e] > 0* **then**
    /* UUtil[] Under-utilised links */
    $UUtil[index1++] \leftarrow$ e
  end
end
$lnk \leftarrow x[i]$

/* Check if the user is assigned to an over-utilised link */
if $(rank[lnk] < 0)$ **then**
  /* Generate a random no. between 0 and 1. */
  $prob \leftarrow$ rand()
  $P_c \leftarrow \frac{abs(rank[lnk])}{U[lnk]}$
  if $(prob < P_c)$ **then**
    /* getnext: Returns an under-utilised link */
    $next \leftarrow$ getnext(); /* defined in algo. 7 */
    /* Check if the user fits; b[i] is incoming traffic of user i */
    $tmp\_util\_next \leftarrow U[next] + b[i]$
    $tmp\_rank\_next \leftarrow IU[next] - tmp\_util\_next$
    if $(tmp\_rank\_next \geq 0)$ **then**
      x[i] $\leftarrow$ next; /* Finally move user */
    end
  end
end

Algorithm 6: Distributed Algorithm: To be executed by each user i.

```
getnext()
/* Return under-utilised link with probability equal to rank.        */
foreach lnk in UUtil do
  | S ← S + rank[lnk]
end
curr_value ← 0
foreach lnk in UUtil do
  | low[lnk] ← curr_value + 1
  | rank_p ← round((rank[lnk] / S) * 100)
  | high[lnk] ← curr_value + rank_p
  | curr_value ← high[lnk]
end
high[lnk] ← 100
prob ← int(100 * rand()) + 1  /* Generate random no.  between 1 and 100  */
foreach lnk in UUtil do
  | if low[lnk] ≤ prob && prob ≤ high[lnk] then
  |  | return lnk
  | end
end
return -1
```

Algorithm 7: getnext(): Return under-utilised link based on rank.

to different access routers. Hence, the scheme of selecting an under-utilised link based on intradomain distance will usually work. However, consider a hypothetical scenario where all users assigned to an over-utilised link are attached to a single access node and there is more than one under-utilised link. If users select an under-utilised link based on intradomain cost, all of them will invariably select the same under-utilised link, the one with the least cost. Understandably utilisation of the links will become highly skewed and the performance will deteriorate consequently. This is another ramification of not being able to globally update the link metrics in a distributed environment. To distinguish this situation we refer to it as the so called *clustered user* scenario. A potential solution is to move users based on the intradomain cost and once the clustered user problem is detected users should be moved based on their link rank. The question is can we detect a scenario where all users start moving to the same under-utilised link? This will not be possible without co-ordination among the users.

In a centralised route control architecture if the *clustered user* scenario occurs, then all users will initially select the least cost under-utilised link. After sufficient number of users gets re-allocated to the link, it will no longer remain under-utilised and the next least cost under-utilised link will become the least cost one. In a way the process of allocating users in the centralised scheme for the clustered scenario will be like filling up a knapsack which is closest to the users. Once the closest knapsack is filled the next closer knapsack is considered.

6.4.3 Hybrid Approach

A distributed heuristic that only considers intradomain cost to choose an under-utilised link does not guarantee that the links will be filled in relation to their rank. Moreover as we have seen it will be susceptible to the clustered user problem. Ideally we would like to fill up the under-utilised links in proportion to their ranks but such a tactic does not consider the intradomain dynamics. In this section we propose a mechanism to select an under-utilised link by unifying the two methods. A link is selected that is based partially on the IGP cost and partially on the link ranks. We call this new mechanism the *hybrid approach* since it is a fusion of the two techniques to select an under-utilised link. The hybrid mechanism has been designed to preempt the occurrence of the *clustered user* problem. In the hybrid approach, a user selects under-utilised links based on the probability of their ranks and assigns a score to each link. The link that is selected first gets the highest score. For example, if there are three under-utilised links the one that gets selected first is assigned a score of 3. The random game is repeated next between the remaining two links and the one that gets selected next is given a score 2 and so on it continues. In the second round, users selects the same set of under-utilised links based on their intradomain cost and assigns a second score to each link. The under-utilised link with the least IGP cost gets the highest score. Continuing with our earlier example the under-utilised link with

the least IGP distance to a user will get a score of 3 and so on the assignment of scores will continue. The final score of a link is the sum of these two scores:

$$T\_score(lnk) \leftarrow score1(lnk, rank) + score2(pathcost(user, lnk))$$

The first component (*score1*) is a function of the link rank and the second component (*score2*) is a function of the IGP cost. To ensure that both of these components have equal contribution in the selection of an under-utilised link the score levels of both are same. The link with the highest final score is the candidate to which users from an over-utilised link will be moved. Now consider an occurrence of the *clustered user* problem, which is all users have the same *score2*. In such a situation *score1* will be used to break the tie. A pseudo-code of the hybrid approach is given in algorithm 8.

```
getnext(user)
/* Copy UUtil to two other arrays                                    */
UUtil1 ← UUtil2 ← UUtil
/* size:  returns the size of an array                               */
size ← size(UUtil); score1 ← score2 ← size
/* Compute score 1                                                   */
/* link-range ():  computes link range, given in algorithm 7         */
link-range(UUtil1)
for j = 1 to size(UUtil) do
    prob ← int(100 * rand()) + 1
    foreach lnk in UUtil1 do
        if UUtil1[lnk] ≠ −1 then
            if low[lnk] ≤ prob && prob ≤ high[lnk] then
                T_score[lnk] ← score1
                − − score1; UUtil1[lnk] ← −1
            end
        end
    end
end
/* Compute score 2                                                   */
src ← i
for j = 1 to size(UUtil) do
    min_cost ← 999; min_lnk ← −1
    foreach lnk in UUtil2 do
        if UUtil2[lnk] ≠ −1 then
            dst ← egress[lnk]; /* Egress node of link lnk            */
            /* pathcost:  returns the IGP cost between two nodes     */
            cost ← pathcost(src, dst)
            if cost < min_cost then
                min_cost ← cost; min_lnk ← lnk
            end
        end
    end
    T_score(min_lnk) ← T_score[min_lnk] + score2 /* Total score     */
    − − score2
    UUtil2[min_lnk] ← −1
end
return (Link with highest total score)
```

Algorithm 8: Hybrid Distributed Approach: Executed by each user i.

6.5 Experimental Results

In this section, we report simulation results to substantiate the theoretical analysis that our proposed distributed schemes indeed emulate the performance of the centralised algorithm (greedy approach). The distributed scheme as one would have observed tries to emulate the behavior of the centralised one. We have already established results for the centralised algorithms for different network environments in the previous chapter. Therefore, a reasonable way of validating the distributed schemes would be compare its performance one to one with those of the centralised schemes. The topologies, traffic traces and settings used in these experiments are, therefore, same as those we had used in the previous chapter.

In this chapter we have effectively proposed three distributed route control techniques. The first one is for a static intradomain network environment, the second for a dynamic network and the third is a fusion of the two. The procedure that decides whether a user will re-assign itself or not is the same for all the three schemes. The difference between the techniques lies in how an under-utilised link is selected. All the three approaches were tested under different topologies and network traffic loads. The experiments are described below. In the first experiment, we test the performance of our distributed algorithm proposed for a static intradomain traffic environment. In the second experiment, we examine the performance of the distributed algorithm proposed for a dynamic environment (i.e. considers IGP cost during egress route selection), using synthetic data. In the third experiment we simulate using real traffic traces. In this experiment we simulate all the three distributed schemes and correlate the results. In the fourth experiment, we analyse the robustness of our approach to sudden reduction in available bandwidth. In the fifth and last experiment of this chapter as well as of this thesis, we investigate the performance of both the centralised and distributed algorithm under high and dynamic traffic load conditions.

6.5.1 Experiment 1: Examining the effects of cross-traffic

This experiment was performed using the simulation program we had developed to test static centralised route control techniques. Details are given in section 5.5.1 of the previous chapter. The simulation program was suitably modified to incorporate the distributed route control technique. Although this simulation does not consider the dynamics of intradomain traffic, nevertheless it measures the performance of our route control mechanism under actual traffic conditions. The experiment was performed using the traffic trace used in experiment 3 of the previous chapter. The number of egress links considered was 4, trace duration was of 1 hour and number of users present was 932. This experiment studies the response of our distributed scheme to different cross traffic scenarios.

Three runs of the experiment were performed. In the first run, no cross traffic was considered. In the second run of the experiment, we considered moderate cross traffic. Finally, in the third run, a very heavy cross traffic was considered. The deviation of the links from their ideal value (i.e ranks) for the *default* case, greedy and the distributed approach for all these three scenarios are given in table 6.2. As we had done in the case of the centralised experiment, to highlight the period-wise performance improvements we plot the deviation of the distributed algorithm using positive and negative bars. A bar is the difference in value of the ranks of the *default* and distributed approach. A positive bar shows the quantum of improvement in the deviation with respect to the *default* approach. Likewise, a negative bar shows the decline in the deviation from that of the *default* approach. The period-wise bars for the three runs when no cross traffic, moderate cross-traffic and heavy cross-traffic were considered are plotted in figure 6.1, 6.2 and 6.3 respectively. We find significant improvement in load balancing when the distributed heuristic is used. Further, we find that the performance of the distributed heuristic is at par to that of the centralised heuristic. A comparison of the statistical properties of the *period-wise improvements* for the distributed and centralised route control techniques are given in table 6.3. We find that the

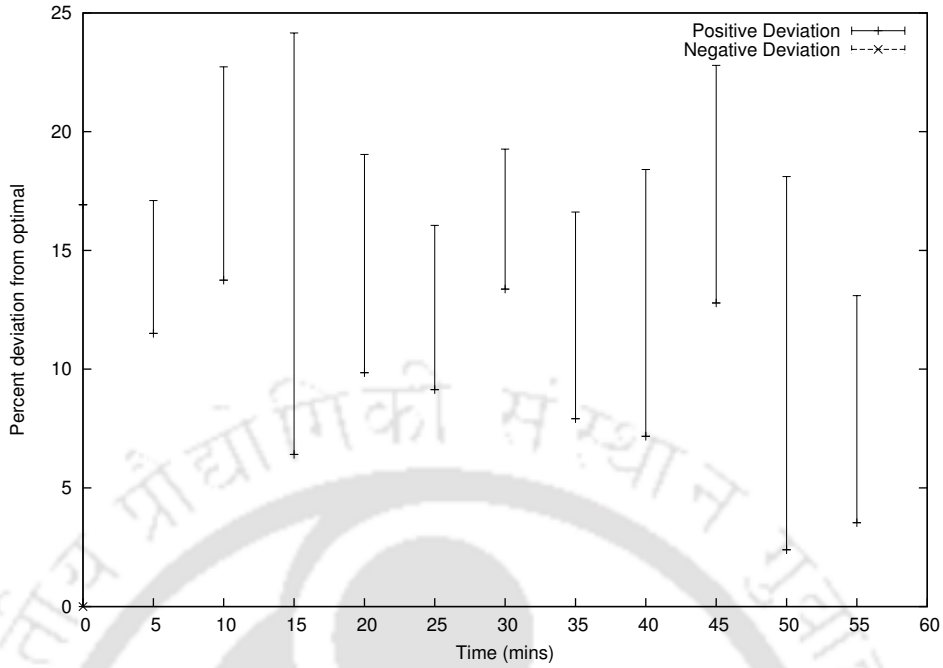
mean and median are almost same while the value of the standard deviation is relatively low. These values suggest that like the greedy approach, the performance benefits for the distributed approach is uniform in all the periods. A comparison of the user re-assignments for the distributed and centralised algorithms are given in table 6.4. In this experiment we find that the number of re-assignments is marginally higher for the distributed case. However, the overall re-assignments for the distributed case is still less, about 2 percent.

Table 6.2 Deviation of link utilisations from their ideal value (in percent)

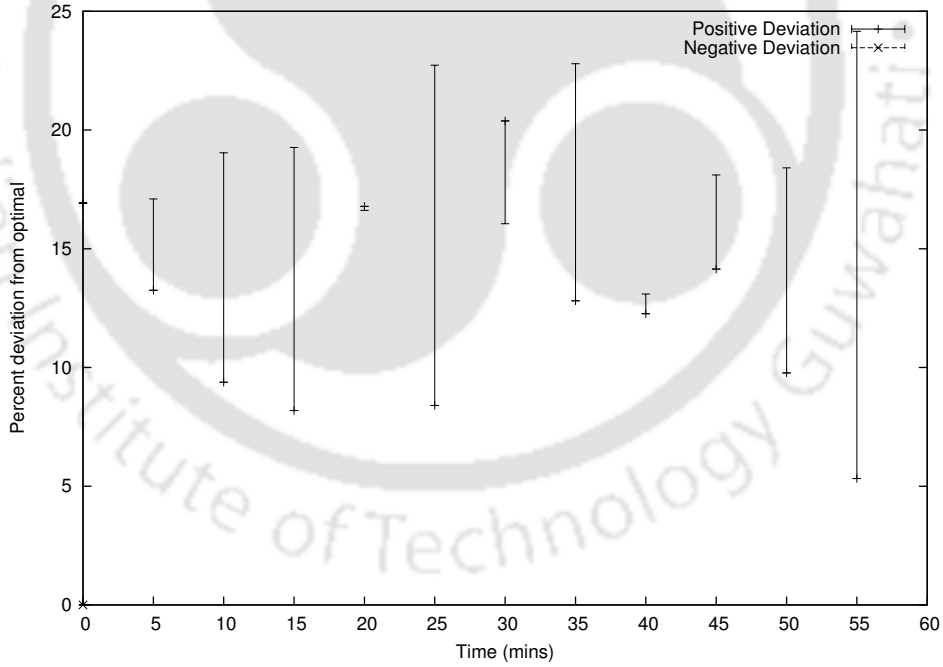
Default	Greedy	Distributed
No Cross Traffic		
18.85	8.89	11.88
Cross Traffic		
17.10	9.77	9.18
Heavy Cross Traffic		
6.47	5.57	5.92

Table 6.3 Improvement of route control scheme in comparison to *default* case (in percent)

Route Control	Mean	Median	Standard Deviation
No Cross Traffic			
Greedy	9.96	9.19	3.78
Distributed	6.97	8.63	6.82
Cross Traffic			
Greedy	7.34	7.09	5.95
Distributed	7.91	8.2	3.94
Heavy Cross Traffic			
Greedy	0.9	-0.36	3.78
Distributed	0.55	1.52	2.89

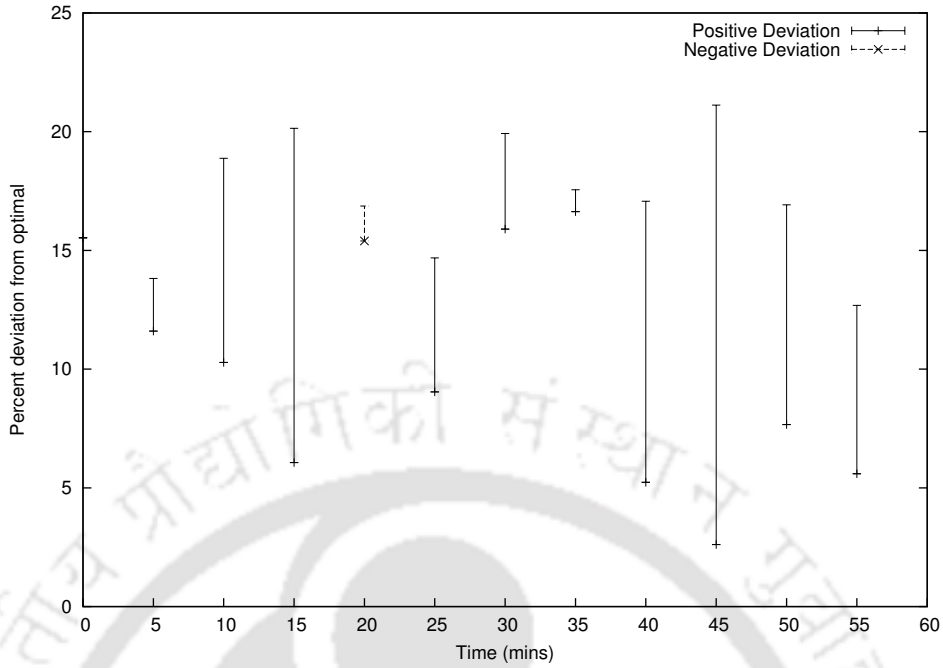


(a) Greedy

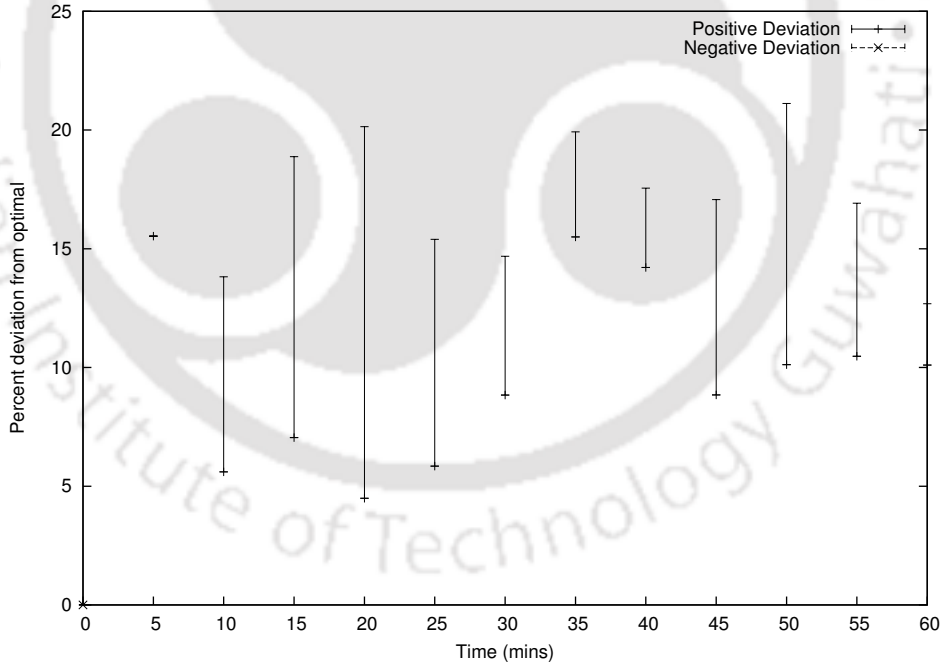


(b) Distributed

Fig. 6.1 Expt 1a. No cross traffic.

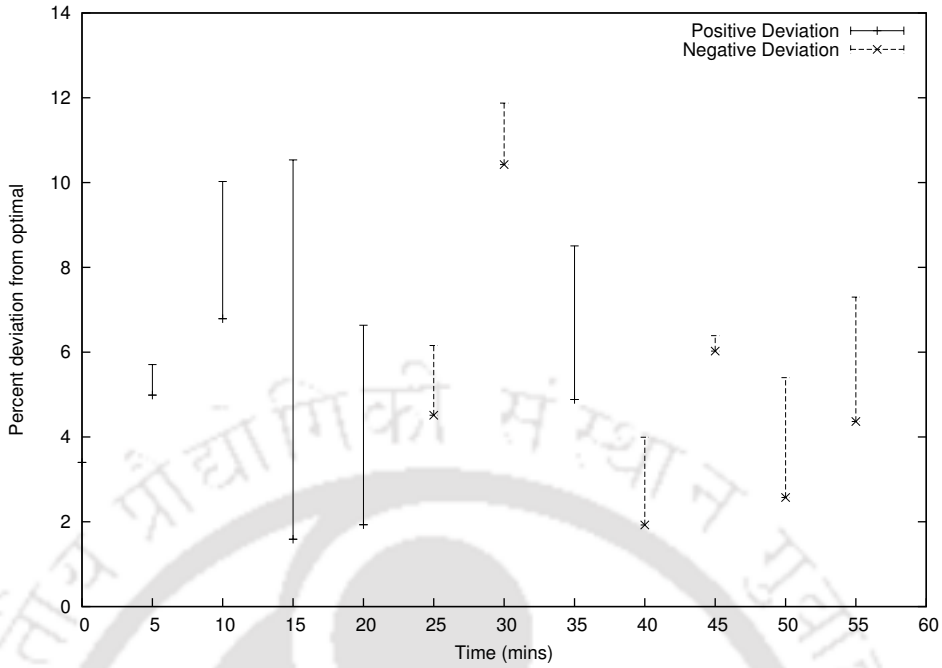


(a) Greedy

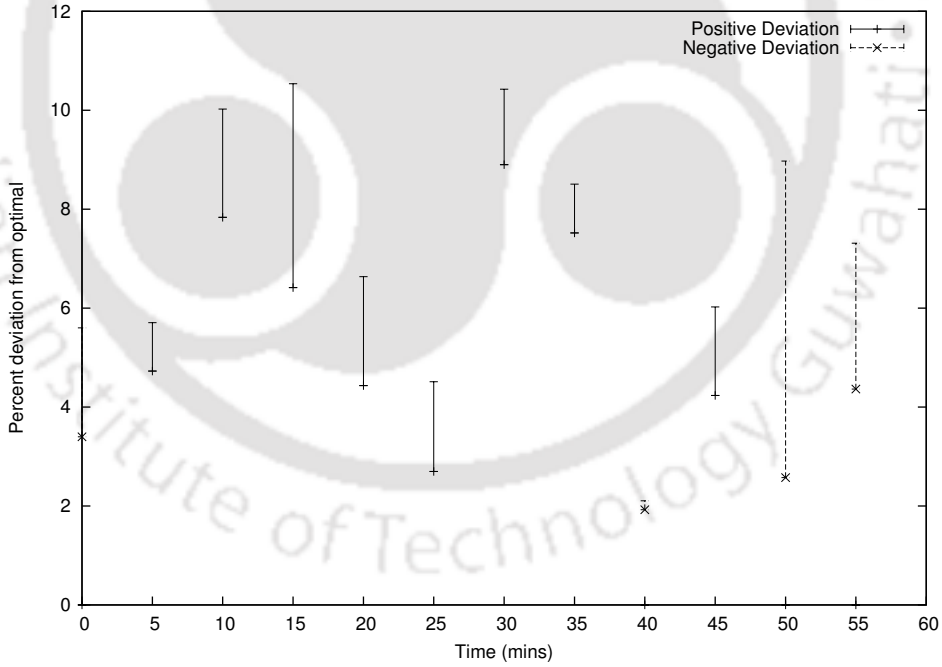


(b) Distributed

Fig. 6.2 Expt 1b. Moderate cross traffic.



(a) Greedy



(b) Distributed

Fig. 6.3 Expt 1c. Heavy cross traffic.

Table 6.4 User Re-assignments (in percent)

Route Control	Mean	Median	Standard Deviation
No Cross Traffic			
Greedy	1.76	1.3	1.83
Distributed	3.38	2.0	3.32
Cross Traffic			
Greedy	0.65	0.3	1.0
Distributed	1.85	2	1.15
Heavy Cross Traffic			
Greedy	1.01	1.1	0.4
Distributed	1.4	1.5	0.77

6.5.2 Experiment 2: Validating using synthetic data

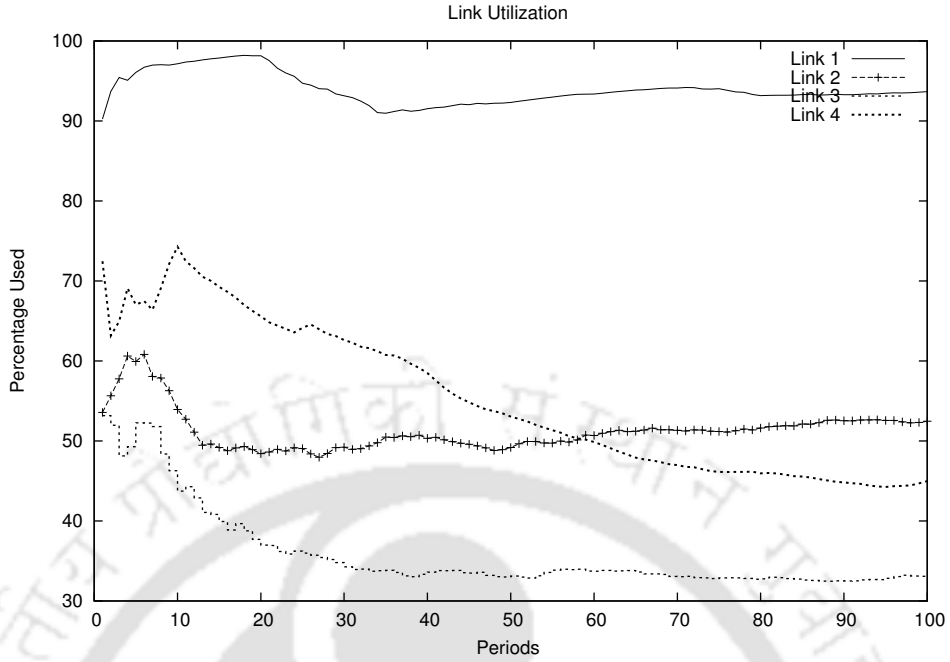
In this experiment, we examine the performance of the proposed distributed algorithm for dynamic network conditions. From among the set of under-utilised links, the algorithm selects one with the least IGP cost (discussed in section 6.4.2). The algorithm was validated using synthetic data. The traffic used in this experiment was synthetically generated using the traffic models of *ns-2* ([5]). In fact, the experimental settings used in this experiment were exactly the same as that of experiment 4 of the previous chapter. The topology considered consisted of 25 nodes, 50 intradomain links and 4 egress links. Capacity of all the egress links were considered to be the same (10 Mbps). The duration of the experiment was 100 periods. In order to compare the output of the simulation with our previously obtained results, the Internet traffic was generated with the same user class settings as that of experiment 4. We had seen earlier, that the average deviation of the links from their ideal when no load balancing techniques were used was 30.67 percent. When the centralised route control technique (greedy) was used the deviations of the links were reduced to 6.03 percent. Plots of the utilisation of the links when no load balancing techniques were used and when the greedy approach was used are reproduced in figure 6.4(a) and 6.4(b) respectively. A plot of the link utilisations, when the distributed route control technique was used is given in figure 6.5(a). We find that the distributed route control technique significantly reduces the deviation of the links. The average deviation of the links is re-

duced to 3.80 percent. Moreover, we find that the plot of the distributed approach and the greedy approach are similar. The overall user assignment for the distributed approach is 4.72 and for the greedy approach it was 4.48 percent. The period-wise re-assignments for the distributed and centralised approaches (for the first 25 periods) are compared using a clustered histogram in figure 6.6. The trend of the re-assignments is also similar.

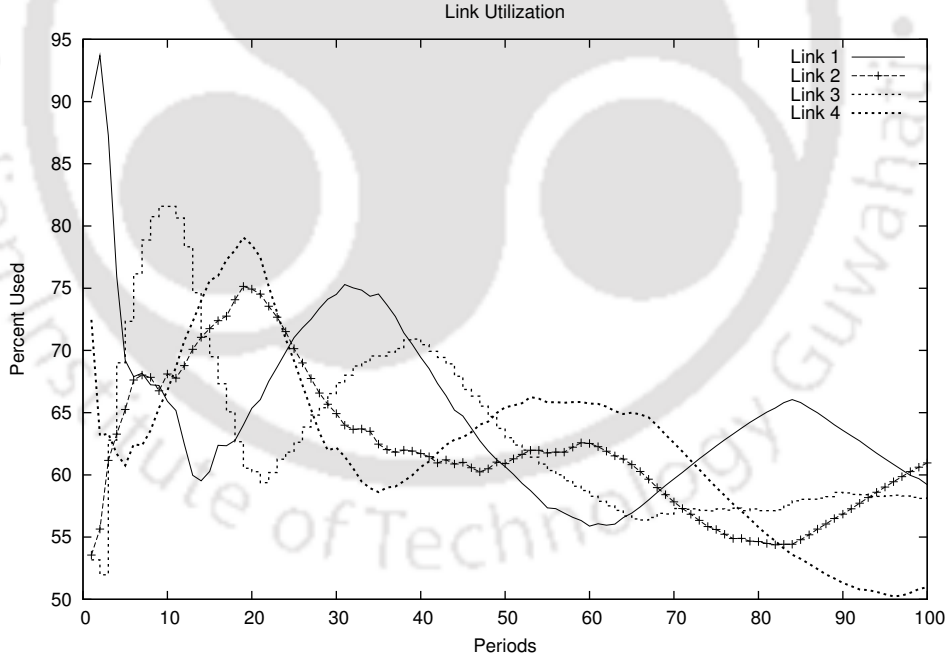
The request-response exchanges between the clients and servers, during the simulation of the distributed approach, were analysed. Table 6.5, shows the traffic profiles for the different schemes. The outgoing and incoming traffic of users in this simulation is comparable to that of the previous experiments. Using the distributed route control technique to load balance Internet traffic resulted in 8.0 percent improvement in round-trip times as compared to the *default* case. Thus, in this experiment too we find that the performance of the distributed route control technique is same as that of the greedy approach in terms of traffic load balancing, traffic re-assignments and improvement in RTTs.

Table 6.5 Expt 2: Characteristics of user traffic.

Experiment	Outgoing Traffic (GB)	Incoming Traffic (GB)	RTT (seconds)
Default	0.17	2.47	990.88
Greedy	0.18	2.43	918.20
Distributed	0.18	2.45	911.02

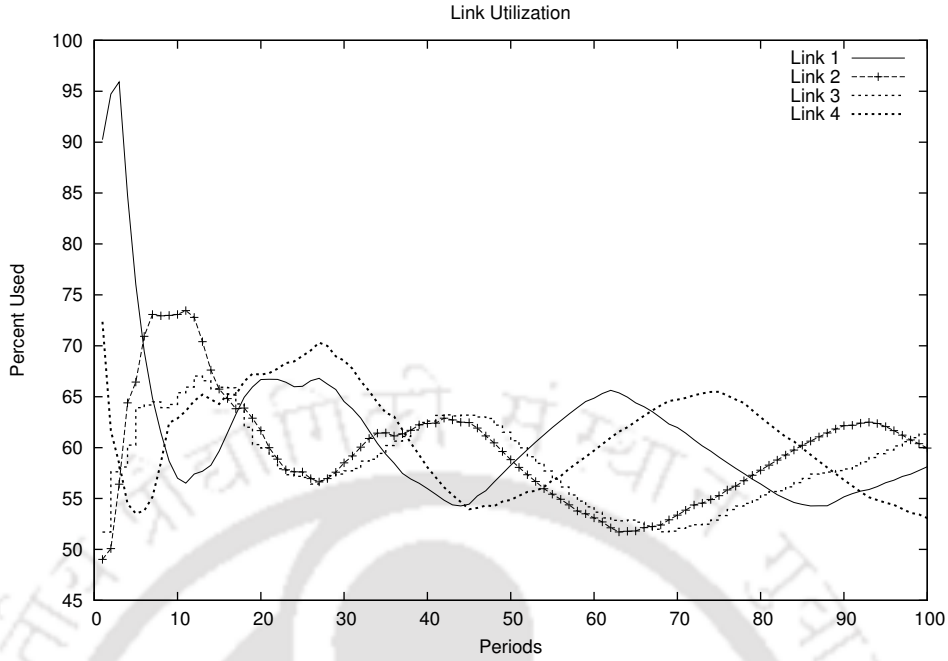


(a) Default



(b) Greedy

Fig. 6.4 Expt 2: Plot of link utilisations (contd).



(a) Distributed

Fig. 6.5 Expt 2: Plot of link utilisations.

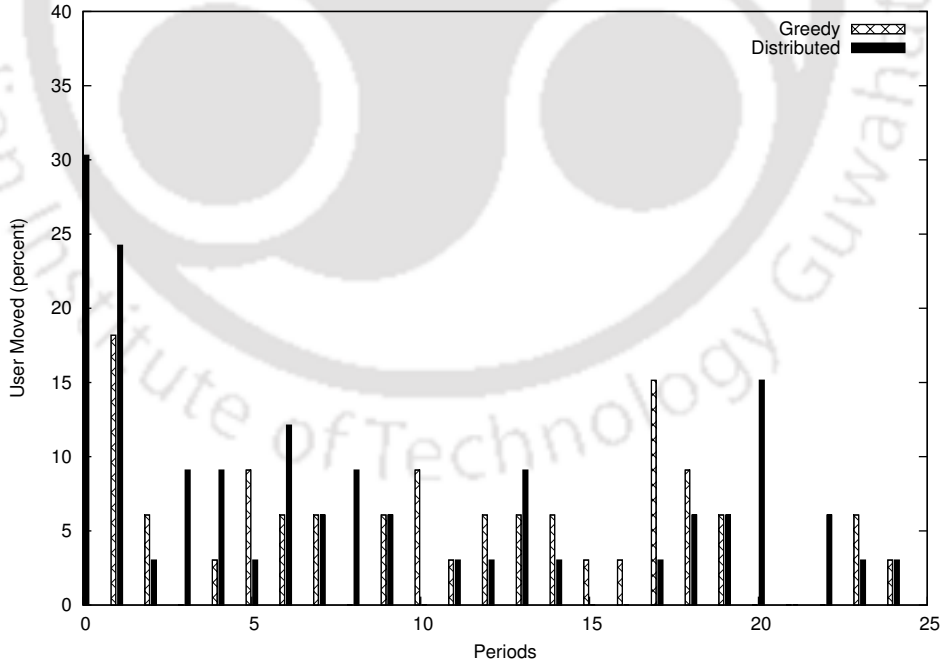


Fig. 6.6 Expt 2: Comparison of re-assignments using clustered histograms.

6.5.3 Experiment 3: Validating using actual data

In this experiment we validate the distributed approach using real traffic traces. The experimental settings and traffic traces used in this experiment were that of experiment 5 of the previous chapter. The topology consisted of 50 nodes and 300 intradomain links. Out of the total nodes, 8 were identified as egress nodes and 17 as access nodes. The traffic trace considered was of 2 hours duration and the number of users present was 1500. With the same experimental setting and traffic trace we simulated all the three distributed route control schemes proposed earlier in sections 6.4.1, 6.4.2 and 6.4.3.

The first simulation of the distributed route control technique considered the rank of a link while selecting an under-utilised link. In figure 6.7(a), we plot the deviation of the links from their ideal value. These values are further compared with the *default* approach and the greedy heuristic using a stacked histogram. The deviation of the links is significantly less when the centralised and distributed route control techniques are used. The average percentage deviations of link utilisations for the *default*, centralised and distributed cases from the ideal value are respectively 14.51, 8.64 and 9.53 respectively. In figure 6.7(b), we plot the re-assignments. In the centralised case, we find that there are a number of periods where the number of user movements is exorbitantly high as compared to other periods. This is because in the centralised approach users are deterministically moved out from an over-utilised links such that sum of the bandwidth of these users equals the rank of the link. To meet this requirement, in periods where the over-utilised link did not have sufficient number of high throughput users, a large number of low throughput users were moved. On the other hand in the case of the distributed approach, the number of re-assignments will depend on the rank of the link as well as count of users present in an over-utilised link. The number of users present in an over-utilised link does not vary widely between periods. Hence, we can see in the figure that the number of users re-assigned is more or less uniform in all the periods. The average percentage of users re-assigned is about 2 percent for the

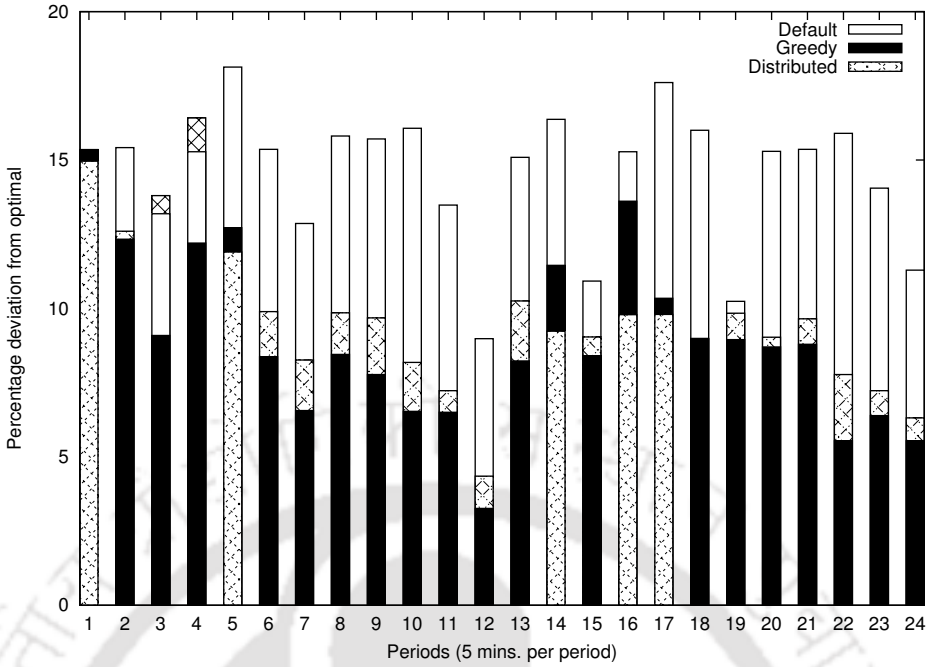
centralised approach whereas it is less than 1 percent for the distributed case.

In the second simulation, the under-utilised links were selected based on the IGP cost. In the third simulation, the hybrid approach was simulated. In figure 6.8, we relate the performance of the two distributed route control schemes. As can be seen from the figure, the distributed route control scheme that considered intradomain cost alone while selecting under-utilised links performs remarkably well. This substantiates our earlier conjecture that if users are distributed evenly, traffic will be disseminated proportionately even if under-utilised links are selected based on intradomain cost rather then selecting them in proportion to their ranks. The overall performance of the hybrid approach is marginally lower. The deviation of the link utilisations from their ideal value for all runs of the experiment are tabularised in table 6.6. The user re-assignments are not plotted for the last two runs of the experiment since the values are almost the same as that of the first experiment.

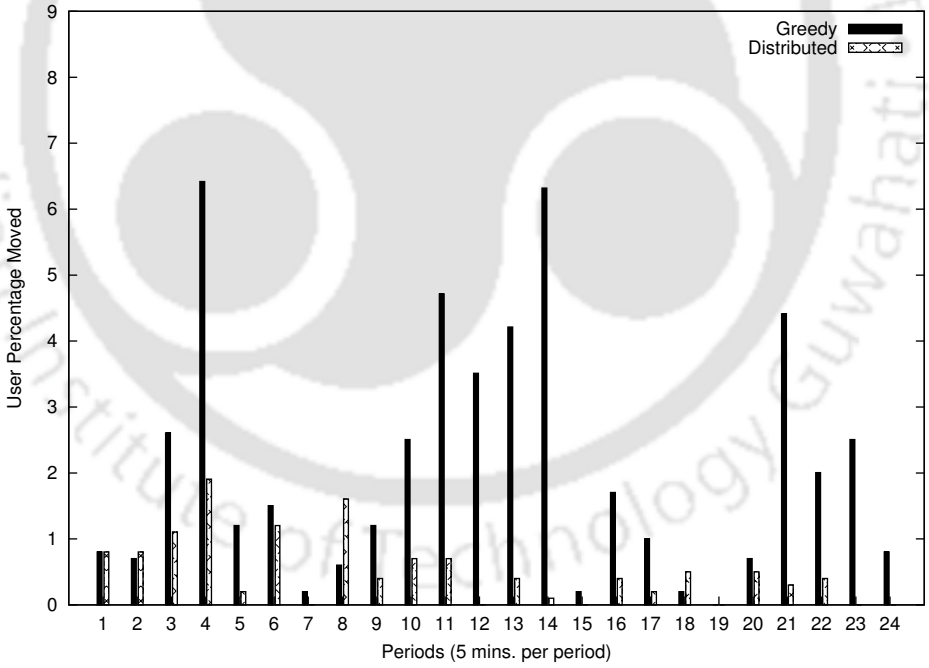
Table 6.6 Deviations from ideal value (in percentage)

Default	Distributed (Static)	Distributed (IGP Cost)	Hybrid
14.51	9.53	9.81	10.41

Although we have proposed different distributed approaches for diverse network situations, the performance of all of them are found to be similar both from a traffic load balancing view point as well as user re-assignments. The similarity in number of users re-assigned is understandable since all the distributed approaches use the same procedure to re-assign a user. One of the main differences between the distributed schemes will be in the intradomain path followed by the user traffic. For different intradomain paths followed, delays incurred by outgoing requests/acknowledgments and latencies of the incoming traffic will differ. To differentiate the distributed schemes, we compared their performance in terms of our third objective, that is the intradomain routes followed by users. One way



(a) Comparison of ranks



(b) Percentage of users re-assigned

Fig. 6.7 Expt. 3: Performance of the distributed approach in static networks.

of comparison is to measure the frequency of different intradomain routes followed by the users. The other way is to compare the cost of the intradomain paths followed by users. When the simulation was run, for each period we output the path followed by a user as well as the cost of the path. The sum of the cost of the intradomain paths for the entire duration of the simulation are tabulated in table 6.7.

Table 6.7 Sum of IGP cost of distributed schemes

Distributed (Static)	Distributed (IGP Cost)	Hybrid
1679	1529.72	1598.22

As we can see from the table, the overall cost of intradomain path followed by the users is 8.9 percent less for the distributed approach that considers intradomain cost while selecting egress routes and 4.8 percent less for the hybrid approach. Thus, while the hybrid approach guarantees that the *clustered user* problem will not occur, its performance will be relatively inferior.

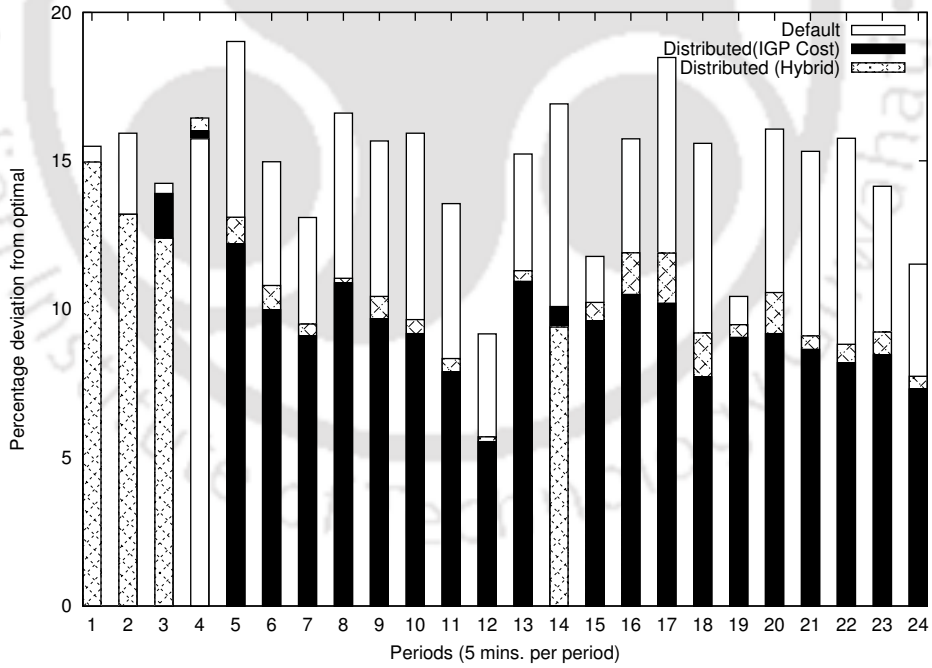
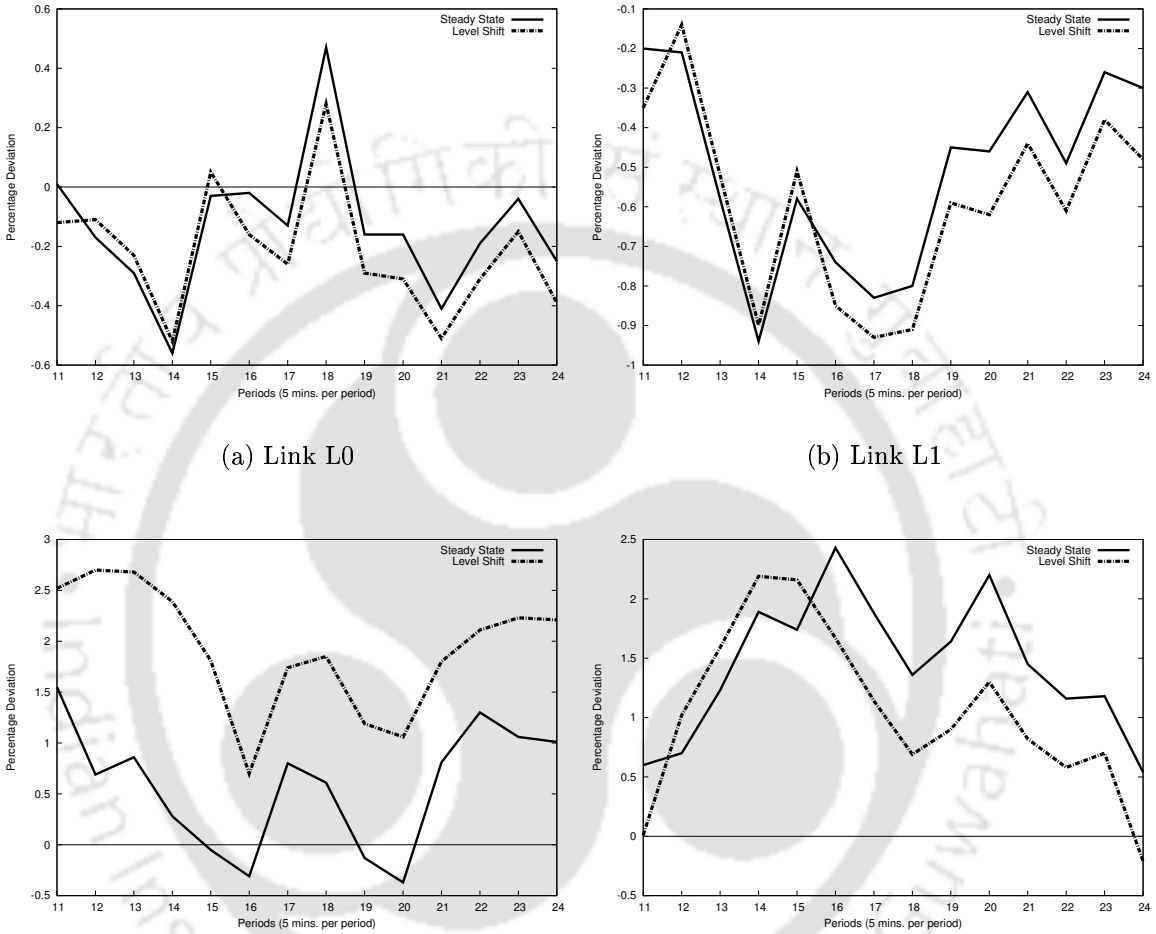


Fig. 6.8 Expt 3: Comparison of distributed approach considering IGP cost and hybrid-distributed approach.

Experiment 4: Examining the robustness of the distributed approach to level-shifts

The implicit assumption made in all our route control schemes is that available bandwidth of the links will not drastically vary between periods. Traffic variability in a time-scale of 5 minutes (the time-period of our proposed route control schemes) is expected to be fairly smooth. However, there can be drastic changes in the available bandwidth due to denial of service attacks, link failures, router flaps or because of other network anomalies (level shift events). In this experiment we examine the responsiveness of our distributed route control scheme to such level shift events. The topology, traffic and reduction in bandwidth considered are exactly same as those of experiment 6 of the previous chapter (section 5.5.4). To create the level shifts, from out of the 8 egress links we reduced the bandwidth of two links, link 4 and link 5, by half. The bandwidth reduction was affected in the middle of the simulation and for a brief duration (periods 12 to 15). In order to ensure that the route control scheme effectively distributes the traffic, we ensured that the total traffic entering the network was less than the total bandwidth available.

The utilisation of the links measured when no level shift events occur was regarded as their steady state values. The comparison of the utilisation of the individual links with their steady state values is given in figure 6.9 and 6.10. The average deviations of the links will obviously deteriorate due to a level shift event for the greedy approach. The average deviation of the distributed approach when no level shift event occurred was 9.81 percent and after the sudden in reduction in available bandwidth we found that the deviations increased to 16.24 percent. The corresponding values for the greedy approach are 9.02 (no level shift events) and 13.12 (after level shift event). The robustness of our distributed scheme to sudden changes in bandwidth is relatively less.



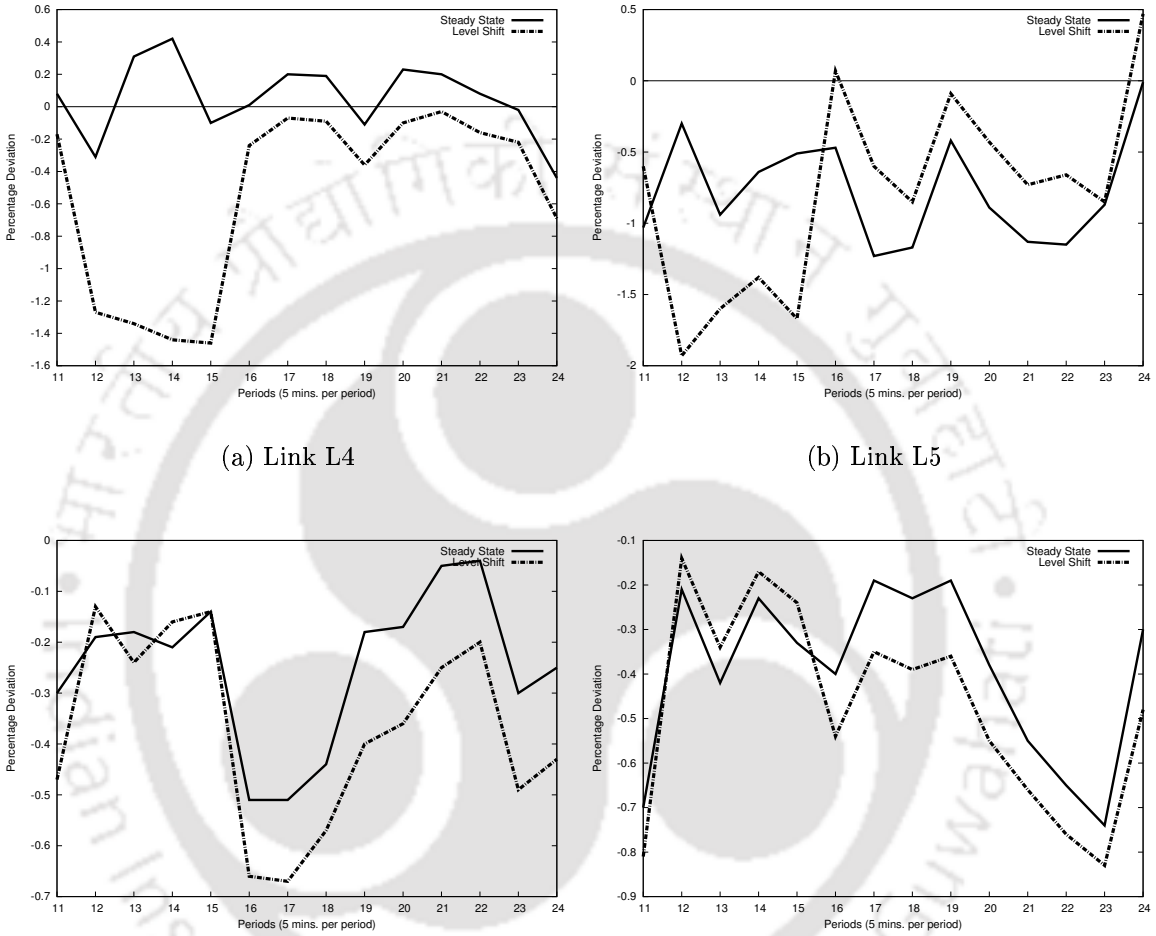
(a) Link L0

(b) Link L1

(c) Link L2

(d) Link L3

Fig. 6.9 Expt 4: Comparison of link deviations with their steady state values for the distributed approach during a level shift event (contd.).



(a) Link L4

(b) Link L5

(c) Link L6

(d) Link L7

Fig. 6.10 Expt 4: Comparison of link deviations with their steady state values for the distributed approach during a level shift event.

6.5.4 Experiment 5: Examining the performance of the greedy and distributed approach under high and dynamic traffic conditions

There are periods when a network can experience very high traffic load situations. During our evaluation of the centralised scheme, we noticed that in very high and dynamic traffic load conditions its performance was relatively lower. In this experiment we test the performance of the centralised and distributed heuristic in a high traffic load scenario. The trace used in this experiment was collected during periods when the network traffic load was dynamic and high. The trace considered was of 5 hours duration. The topology that we used in this simulation had 50 nodes and 50 intra-domain links. Out of the 50 nodes, 8 were egress nodes, 17 access nodes and the rest were core nodes. The total number of users present in the trace was about 1000. The total traffic downloaded was 20 gigabytes and the outgoing traffic was 2.2 gigabytes. Users were uniformly distributed among the access nodes. The user re-assignments for both the schemes are shown in figure 6.11(b). The output plotted is for the first two hours only. As was in the case of the third experiment, the user re-assignments for the distributed scheme are lower in all the periods. In this experiment we found that the greedy approach performed better than the *default case* in only 40 percent of the periods. Interestingly in this experiment, the distributed approach despite of its lower user movements does not fare poorly. On the contrary the distributed approach performed better than the centralised approach in terms of load balancing (figure 6.11(a)). The average percentage deviations from the ideal value for the *default*, greedy and distributed approach are respectively 9.30, 8.40 and 7.26 percents.

As one can see the performance improvements of this experiment are modest. To investigate further we compared the traffic trace used in this experiment with those traces used in our earlier experiments. For example, if we compare the traffic trace of this experiment with that of experiment 3, the first observation is that the deviation of the link utilisations from their ideal value, when no load balancing scheme was used, is relatively less. The av-

erage deviation of the links for the *default* case was 14.51 percent in experiment 3, whereas in this experiment it is only 9.30 percent. This means in this experiment, utilisation of all the links are closer to their available bandwidth. From a traffic load balancing context it means the links have almost reached their threshold values and there is lesser room to maneuver. Hence, we find the performance of our load balancing schemes is relatively lower.

The other point we examine is why the load balancing performance of the greedy approach is lower as compared to the distributed approach, in spite of the re-assignments being higher in the case of the greedy approach. Considering the fact that the deviation of the *default* case in this experiment is lower as compared to the third experiment, one would expect the number of user re-assignments to be proportionately lower. However, if we relate the user movements for the greedy heuristic of these two experiments, we find that the percentage of users re-assigned is higher in this experiment. To highlight the effect of dynamic traffic condition on link utilisations, we plot the rank of one of the links in figure 6.12. The x-axis represents the ideal utilisation. As can be seen from the figure, the *default* link utilisation intersects the x-axis several times. To paraphrase this means even when no load balancing techniques were used, utilisation of the link varied from over-utilisation to under-utilisation and vice-versa frequently. The greedy approach tried to aggressively adjust its traffic in accordance with the *default* utilisation. This explains for the higher user re-assignments. Further because of this same reason we can see in the figure, the greedy approach grossly over-estimates during periods when the utilisation of the link fluctuates. On the other hand in the case of the distributed approach, the number of re-assignments does not depend on the incoming traffic but on the absolute value of rank and users present in the over-utilised link. Hence we find that fluctuation in re-assignments is relatively less for the distributed approach. The use of a probability game in the distributed scheme will ensure that there will be some dispersion of the re-assignments. The less aggressive approach of traffic re-assignment in the case of the distributed techniques resulted in better

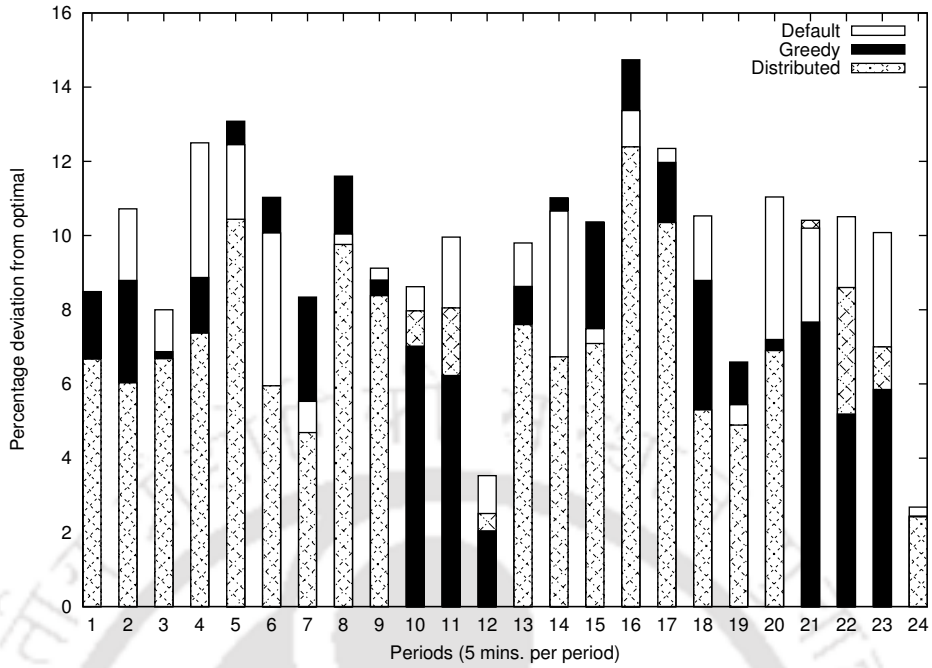
smoothening out of the fluctuations of link utilisations. The difference in deviation of the link utilisations when our proposed route control schemes were used to that of the *default* case is computed and its statistical properties are listed in table 6.8. A positive value indicates the quantum of improvement in the deviation, when our route control mechanism is used, with respect to that of the *default* approach. Similarly a negative value would indicate the degradation in performance. As can be seen, all the numerals in the table have positive values. However, the values are relatively small for this experiment.

Table 6.8 Difference in link ranks (as compared to the *default* case in %)

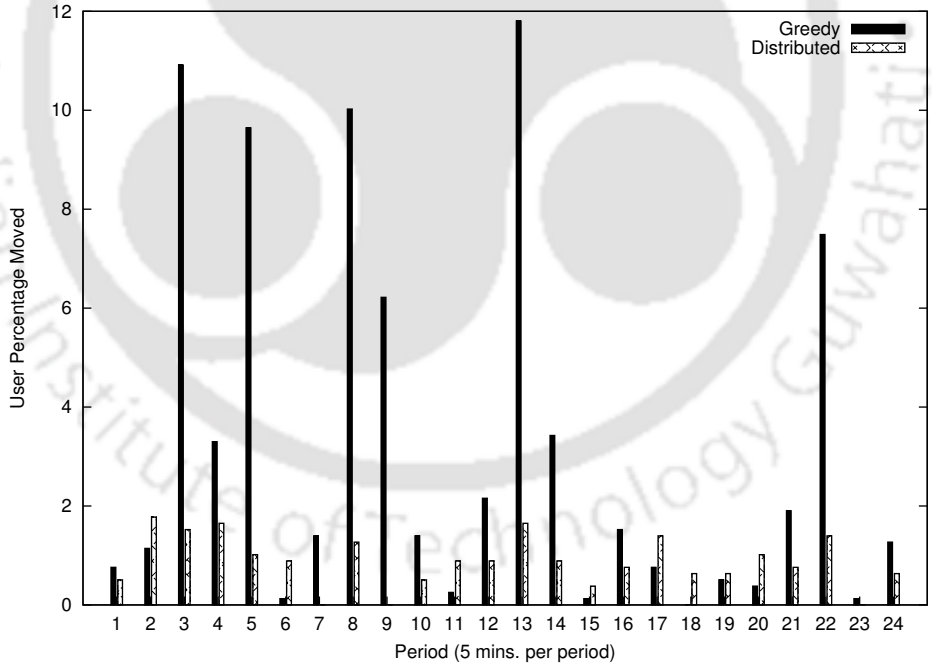
Route Control	Mean	Median	Standard Deviation
Experiment 3			
Greedy	5.87	6.57	2.34
Distributed	4.98	5.71	2.61
Experiment 5			
Greedy	0.9	1.15	2.21
Distributed	2.04	1.87	1.67

6.5.5 Conclusion

To make our route control techniques more practical and scalable, in this chapter we proposed a distributed version of our route control scheme. We focused on adapting the centralised algorithms proposed in chapter 5, in particular the greedy approach. We establish that the *expected* performance of our proposed distributed route control schemes is comparable to those of the centralised ones. In order to validate the distributed route control schemes, we basically repeated the experiments performed for the centralised algorithm and compared the results one-to-one. After each simulation we compared the distributed algorithm with the centralised greedy approach. We examined the performance of our distributed route control scheme both in a static intradomain traffic environment as well as in a dynamic network environment. The simulations were performed using synthetic as well as actual traffic traces. In all the experiments that were performed, we found that the performance of the distributed schemes is comparable to the centralised techniques in



(a) Comparison of ranks



(b) Percentage of users re-assigned

Fig. 6.11 Expt 5: Performance comparisons under high traffic load scenario.

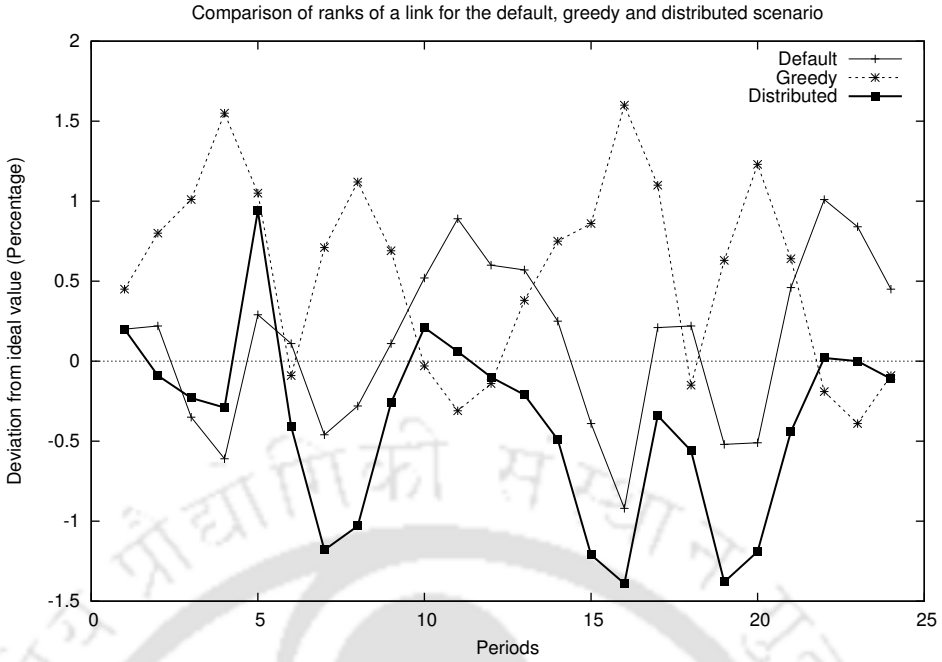


Fig. 6.12 Expt 5: Plot of rank of a link in a dynamic network traffic scenario.

terms of traffic load balancing, traffic re-assignments and improvement in RTTs.

In this chapter we proposed two distributed schemes for selecting egress routes in a dynamic network environment. The first one selects an egress route based on the IGP cost between the user and the egress router. The second one is an hybrid approach, the selection of an egress route partially depends on the IGP cost and partially on the link rank. The performance of the hybrid approach was found to be marginally lower. The hybrid approach was proposed to avoid the so called *clustered user problem*. However, in general during our experiments we found that the *clustered user problem* is unlikely to occur in actual networks. Therefore, the distributed scheme that selects egress routes based on the IGP cost will work efficiently in most networks. We concluded the chapter by investigating the performance of the centralised and distributed route control schemes in high and dynamic traffic conditions. Interestingly, we found that the distributed techniques performed better than the centralised ones both in terms of traffic load balancing as well as re-assignments in such network conditions.

Chapter 7

Conclusions and Future Work

A lot of research work has been done to optimise the use of Internet resources. Most of these solutions are beset with the problem that it requires cooperation from other competing networks, require Internet-wide infrastructure, or assumptions made about network traffic are not realistic. This leads to implementation complexities and as a result the proposed solutions do not get implemented in actual networks. In this work we have proposed simple, light-weight and practically viable solutions. Our proposed schemes can be deployed with minimal changes to existing networks. The techniques proposed do not require changes to existing routing protocols or standards.

7.1 Contributions

In this thesis we have demonstrated how a multihomed end network can benefit by redistributing its traffic. Most of the works on traffic engineering mainly consider the outgoing traffic. Controlling the incoming traffic of a network is difficult since it requires influencing the behavior of remote destinations. In this work our target network is *access networks*, where downloads are much higher than upload. In this dissertation we have sought to redistribute the incoming traffic of a network by regulating the corresponding outgoing traffic. The solutions proposed do not require cooperation from uplink providers. The

other interesting point of our approach is that we do not make any assumptions about the traffic or state of the network.

7.1.1 Intelligence at the Edge

The main design principle that we have followed is to keep the core of the Internet simple and allow intelligence to reside at the edge of the network. The issues that can affect Internet performance can be categorised as *local* and *remote*. Addressing the remote issues will usually require network wide upgradation and management, while local issues can be addressed by locally updating the system. The key local issue that we address in this work is to optimally utilise the resources at the disposal of the network. We identify that the first few hops that connects an end network with its ISP (*connecting link*) is the primary resource that needs to be optimised. The connecting link that we discuss in this dissertation is different from the last mile connectivity of a network. We believe that this notion of connecting link can also be applied to other areas of interdomain traffic engineering.

To estimate the performance metrics of the *connecting link*, we need to locate the bottlenecks on Internet paths that connects an end network with its ISP and then measure its available bandwidth. A major challenge in interdomain traffic engineering is to correctly estimate link metrics in the presence of uncertainties. In this work we survey a large array of tools available in the public domain for measuring available bandwidth. We use *pathneck*, a single-ended, open source tool that can both locate congestion points as well as estimate the bandwidth of the bottleneck path. We probed a large number of Internet paths using this tool. We established that such single-ended tools conflate the delays of the reverse path too and in access networks it effectively measures the metrics of the incoming traffic. The performance of such tools is highly dependent on the configuration parameters, and, due to error in the parameters, or otherwise, the results can be unrealistic at times. Hence, we developed a simple mechanism to ensure that the results reported by pathneck do not

stray beyond an upper limit.

7.1.2 Benefits of Multihoming

Although multihoming has been traditionally associated with resilience, today multihomed networks employ a variety of route control techniques called *smart* or *intelligent* routing to improve their Internet experience. A number of commercial tools are available. However, not much is known about the performance and working of the commercial multihoming route control techniques. In the literature a number of multihoming route control techniques have also been proposed. These proposals have quantified the benefits of multihoming by load balancing traffic among *well-chosen* ISPs or by grouping ISPs based on their geographical proximity. In this work we re-distribute the traffic among *available* egress links based on their current traffic load carrying capacity. Since we concentrate on using the available resources it makes our work more realistic and practical. To evaluate our approach for the general case, where an end network can be multihomed to K ISPs, we deployed a prototype on a 3-multihomed end network. The improvements in round-trip times were significant. We show that performance improves by 10% when two ISPs are used and by 16% when three ISPs are used. Further we show that our route control strategy can be applied transparently without disturbing the existing network setup. The cost incurred to re-distribute the traffic is small, ranging from 0.5 to 2% of the total number of users.

7.1.3 Scaling the Benefits of Multihoming

We extended our route control technique to include multihomed, stub ASes like that of medium ISPs and corporate networks. Again the costly resource that needs to be optimised here is their interdomain connectivity. However, path to an egress router from a node can be several hops away and the intradomain traffic can have a significant contribution on the overall latencies. Further as the size of a network increases the traffic handled by it will be higher and as a result measurement errors are likely to be more prominent. We

model the problem in such a way that the traffic is re-distributed based on the current load of a link, cross traffic present and intradomain cost. At the same time we have tried to minimise the re-assignment of traffic. The modeling allows a node to search for available bandwidth on alternate paths and thereupon load balance the traffic. The design also ensures that utilisation of the links are fairly balanced even if the network measurements are approximate. We proposed a number of heuristics. Simple analysis shows that they are near optimal in a static environment. Extensive simulations using both synthetic as well as actual Internet traces demonstrate that in a dynamic environment the heuristics give significant performance benefits. The heuristics proceed in periods and the amount of traffic re-allocated in each period is uniform. The other advantage of our route control techniques are that they preempt the occurrence of traffic oscillations.

7.1.4 Distributed Multihoming Route Control

The problem of re-assigning traffic is best considered at the global level by a manager that can see the traffic dynamics of the whole network as well as the current user assignments. A natural continuation of our work would be extending our approach to a distributed environment where each user makes its own decision. Leveraging upon our solutions for the centralised approach we focus on customising these techniques for a distributed approach. Using analysis we show that the expected performances of the distributed schemes are comparable to those of the centralised ones. Empirical results show that the performance of the distributed schemes match the centralised techniques in terms of traffic load balancing, traffic re-assignments and improvement in RTTs.

In this thesis we have proposed simple, low-cost route control techniques to load balance traffic. The schemes were tested under numerous traffic load conditions and topologies. We found that our techniques can achieve good load balancing of the links with small traffic re-assignments. The schemes do neither require coordination between its uplink network

nor changes to existing standards and protocols. Hence the schemes can be easily deployed by an end network. The proposed techniques can scale to large-size end networks.

7.2 Future Work

In the following sections, we discuss key issues that were not addressed in this dissertation.

7.2.1 Correlating Traffic Re-assignment with Load Balancing

During our extensive simulation we observed two general trends. The first is higher the traffic load imbalance for the *default case*, greater is the performance improvement of our load balancing schemes. The second general observation is performance improvement is proportional to the traffic re-assigned. However, we observe that there are exceptions. The performance of route control techniques also depends on the network state. Hence, we could not put forward a single heuristic that gives the best performance in all traffic conditions. We need to further study the correlation between load balancing, traffic reassignment and network state. This will help in further integration of our techniques and present a single technique that will give the best performance irrespective of the traffic conditions.

7.2.2 Global Effects of Local Optimisation

In ([56]) it has been shown that *smart-routing* users can coexist well with very little interference to other users. A related issue that needs to be addressed is the interaction between multiple networks that use our route control technique. Let us consider a scenario where two networks connected using multiple links tries to optimise their incoming traffic. Since we focus on re-distributing traffic from a congested path and decouple the incoming and outgoing bandwidth, the goal of the two networks will not conflict. We neither try to influence the routing decisions of the remote destination nor do we interfere with the route updates (in the NAT-based model). Hence, our approach will not cause instability in the core of the Internet. However, we need to study more carefully the scenario where a

large number of networks spread across the Internet use our route control technique. We also need to study how our approach coexists with different players that do not use traffic engineering tools or have conflicting goals (for example tries to optimise their outgoing traffic).

7.2.3 Reducing overheads of BGP

In this dissertation we have maintained that given the present networking protocols and standards, the most practical way of regulating incoming traffic would be to use NAT-based solutions. However, for the solutions to scale to larger networks we have proposed tweaking the configuration of BGP and proposed a new model called *selective sub-prefix advertisement*. The changes we have suggested to the BGP policy require frequent withdrawal and re-advertisement of routes. But our proposal tries to ensure that *route flapping* does not occur due to these frequent updations. Nevertheless, we need to study carefully the route propagation triggered by a re-advertisement as well as its effect on the interdomain routing of other neighbouring domains. We can also look at the possibility of forging cooperation between an end network and its uplink providers so that smaller IP prefixes can be used for advertisement. This will in turn help in reducing the granularity of load balancing and thereby further improve performance.

References

[1] FatPipe. "http://www.fatpipeinc.com".

[2] Internap Network Services Corp.:Flow Control Platform[online]. "http://www.internap.com/product/technology/fcp/page1533.html".

[3] . Internet World Stats[online]. http://www.internetworldstats.com/stats.htm, .

[4] Nortel Networks. "http://www.nortelnetworks.com".

[5] ns-2, Network Simulator version 2. http://www.isi.edu/nsnam/ns/.

[6] . OSPF Design Guide[online]. http://www.cisco.com/warp/public/104/1.html, .

[7] Perl [online]. http://www.perl.org/.

[8] Radware: Linkproof. "http://www.rad-direct.com/datasheet/linkproof.pdf".

[9] Rether Networks Inc. "http://www.rether.com".

[10] RouteScience. "http://www.routescience.com".

[11] Standard Performance Evaluation Corporation, SPECweb2005[online]. http://www.spec.org/web2005.

[12] tcpdump - Dump Traffic on a Network[online]. http://www.tcpdump.org/.

[13] tcpslice: A Tool for Extracting Portions of Packet Trace Files. http://www.tcptrace.org/.

[14] The Multi Router Traffic Grapher (MRTG) [online]. http://www.mrtg.com.

[15] Traceroute [online]. Available: ftp://ftp.ee.lbl.gov/traceroute.tar.z.

[16] Performance Measurement Tools Taxonomy [online]. http://www.caida.org/tools/taxonomy/performance.xml, March 2007.

[17] A. Feldmann and A. G. Greenberg and C. Lund and N. Reingold and J. Rexford and F. True. Deriving Traffic Demands for Operational IP Networks: Methodology and Experience. In *SIGCOMM*, pages 257–270, 2000.

[18] A. Medina and A. Lakhina and I. Matta and J. Byers. BRITE: An Approach to Universal Topology Generation. In *Proceedings. Ninth International Symposium on Modeling, Analysis and Simulation of Computer and Telecommunication Systems*, pages 346–353, 2001.

[19] S. Agarwal, C. Chuah, S. Bhattacharyya, and C. Diot. Impact of BGP Dynamics on Intra-Domain Traffic, 2004.

[20] S. Agarwal, Chen-Nee Chuah, and R.H. Katz. Robust Interdomain Policy Routing and Traffic Control. *Open Architectures and Network Programming, 2003 IEEE Conference on*, pages 55–64, 4-5 April 2003.

[21] A. Akella, B. Maggs, S. Seshan, and A. Shaikh. On the Performance Benefits of Multihoming Route Control. *Networking, IEEE/ACM Transactions on*, 16(1):91–104, Feb. 2008.

[22] A. Akella, S. Seshan, and A. Shaikh. An Empirical Evaluation of Wide-Area Internet Bottlenecks. In *IMC '03: Proceedings of the 3rd ACM SIGCOMM conference on Internet measurement*, pages 101–114, New York, NY, USA, 2003. ACM.

[23] David G. Andersen, Hari Balakrishnan, M. Frans Kaashoek, and Robert Morris. Resilient Overlay Networks. In *Symposium on Operating Systems Principles*, pages 131–145, 2001.

[24] G. Apostolopoulos, D. Williams, S. Kamat, R. Guerin, and A. Orda. QoS Routing Mechanisms and OSPF Extensions, August 1999.

[25] S. Arora and B. Brinkman. A Randmoized Online Algorithm for Bandwidth Utilization. *Journal of Scheduling*, 7(3):187–194, May 2004.

[26] Yossi Azar, Andrei Z. Broder, and Anna R. Karlin. On-line Load Balancing. *Theoretical Computer Science*, 130(1):73–84, 1994.

[27] Yossi Azar, Bala Kalyansasundaram, Serge A. Plotkin, Kirk Pruhs, and Orli Waarts. On-line Load Balancing of Temporary Tasks. *J. Algorithms*, 22(1):93–110, 1997.

[28] B. Fortz and M. Thorup. Internet Traffic Engineering by Optimizing OSPF Weights. In *INFOCOM (2)*, pages 519–528, 2000.

[29] B. Quoitin and S. Uhlig and C. Pelsser and L. Swinnen and O. Bonaventure. Interdomain Traffic Engineering with BGP. *IEEE Communications Magazine*, 2003.

[30] B. R. Greene and P. Smith. *Cisco ISP Essentials*. Cisco Press, 2002.

[31] S. Bellovin. A Best-Case Network Performance Model. Tech. Rep., ATT Research, Feb. 1992.

[32] O. Bonaventure, C. Filsfils, and P. Francois. Achieving sub-50 Milliseconds Recovery upon BGP Peering Link Failures. *IEEE/ACM Trans. Netw.*, 15(5):1123–1135, 2007.

[33] N. Brownlee and K. C. Claffy. Understanding Internet Traffic Streams: Dragonflies and Tortoises. *Communications Magazine, IEEE*, 40(10):110–117, Oct 2002.

[34] P. Ramanathan C. Dovrolis and D. Moore. What do Packet Dispersion Techniques Measure? In *INFOCOM 2001*, Jan 2001.

[35] Jin Cao, W.S. Cleveland, Yuan Gao, K. Jeffay, F.D. Smith, and M. Weigle. Stochastic Models for Generating Synthetic HTTP Source Traffic. *INFOCOM 2004. Twenty-third AnnualJoint Conference of the IEEE Computer and Communications Societies*, 3:1546–1557 vol.3, 7-11 March 2004.

[36] Robert L. Carter and Mark E. Crovella. Measuring Bottleneck Link Speed in Packet-Switched Networks. *Perform. Eval.*, 27-28:297–318, 1996.

[37] K. Chandrayana, R. M. Karp, M. Roughan, S. Sen, and Y. Zhang. Search Strategies in Inter-Domain Traffic Engineering. International Computer Science Insitute, http://www.icsi.berkeley.edu/cgi-bin/pubs/publication.pl?ID=000171.

[38] R. Chinchilla, J. Hoag, D. Koonce, H. Kruse, S. Ostermann, and Y. Wang. Characterization of Internet Traffic and User Classification: Foundations for the Next Generation of Network Emulation. In *Proceedings of the 10th International Conference on Telecommunication Systems, Modeling and Analysis*, 2002.

[39] Seung-Hwa Chung, Y.J. Won, D. Agrawal, Seong-Cheol Hong, James Won-Ki Hong, Hong-Taek Ju, and Kihong Park. Detection and Analysis of Packet Loss on Under-utilized Enterprise Network Links. *End-to-End Monitoring Techniques and Services, 2005. Workshop on*, pages 164–176, May 2005.

[40] D. Clark. The Design Philosophy of the DARPA Internet Protocols. In *SIGCOMM '88: Symposium proceedings on Communications architectures and protocols*, pages 106–114, New York, NY, USA, 1988. ACM.

[41] M.E. Crovella and A. Bestavros. Self-similarity in World Wide Web Traffic: Evidence and Possible Causes. *Networking, IEEE/ACM Transactions on*, 5(6):835–846, Dec 1997.

[42] D. Allen. NPN: Multihoming and Route Optimization:Finding the Best Way Home. *Network Magazine*, page , February 2002.

[43] D. Awduche and A. Chiu and A. Elwalid and I. Widjaja and X. Xiao. Overview and Principles of Internet Traffic Engineering. In *RFC 3272*, May 2002.

[44] D. Kaitz and K. Kompella and D. Yeung. Traffic Engineering Extensions to OSPF Version 2. In *RFC 3630*, September 2003.

[45] E. Rosen and A. Viswanathan and R. Callon. Multiprotocol Label Switching Architecture. In *RFC 3031*, January 2001.

[46] W. Fang and L. Peterson. Inter-AS Traffic Patterns and Their Implications. *Global Telecommunications Conference, 1999. GLOBECOM '99*, 3:1859–1868 vol.3, 1999.

[47] N. Feamster, J. Borkenhagen, and J. Rexford. Guidelines for Interdomain Traffic Engineering. *SIGCOMM Comput. Commun. Rev.*, 33(5):19–30, 2003.

[48] A. Feldmann, A. Feldmann, A. C. Gilbert, and W. Willinger. Data Networks as Cascades: Investigating the Multifractal Nature of Internet WAN Traffic. *SIGCOMM Comput. Commun. Rev.*, 28(4):42–55, 1998.

[49] Anja Feldmann, Anna C. Gilbert, Polly Huang, and Walter Willinger. Dynamics of IP Traffic: A Study of the Role of Variability and the Impact of Control. *SIGCOMM Comput. Commun. Rev.*, 29(4):301–313, 1999.

[50] S. Floyd and V. Paxson. Difficulties in Simulating the Internet. *Networking, IEEE/ACM Transactions on*, 9(4):392–403, Aug 2001.

[51] Sally Floyd and Van Jacobson. Random Early Detection Gateways for Congestion Avoidance. *IEEE/slash ACM Transactions on Networking*, 1(4):397–413, 1993.

[52] B. Fortz, J. Rexford, and M. Thorup. Traffic Engineering with Traditional IP Routing Protocols, 2002.

[53] M. R. Garey and D. S. Johnson. " Strong " NP-Completeness Results: Motivation, Examples, and Implications. *J. ACM*, 25(3):499–508, 1978.

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

[55] T. Goff, J. Moronski, D. S. Phatak, and V. Gupta. Freeze-TCP: A True End-to-End TCP Enhancement Mechanism for Mobile Environments. In *INFOCOM (3)*, pages 1537–1545, 2000.

[56] D. K. Goldenberg, L. Qiuy, H. Xie, Y. Richard Yang, and Y. Zhang. Optimizing Cost and Performance for Multihoming. *SIGCOMM Comput. Commun. Rev.*, 34(4):79–92, 2004.

[57] F. Guo, J. Chen, W. Li, and T. Cker. Experiences in Building A Multihoming Load Balancing System. In *INFOCOM 2004.*, 2004.

[58] Anupam Gupta, Martin Pál, R. Ravi, and Amitabh Sinha. Boosted Sampling: Approximation Algorithms for Stochastic Optimization. In *STOC '04: Proceedings of the thirty-sixth annual ACM symposium on Theory of computing*, pages 417–426, New York, NY, USA, 2004. ACM Press.

[59] H. Wei and S. Tsao and Y. Lin. Assessing and Improving TCP Rate Shaping over Edge Gateways. In *IEEE Transactions on Computers*, page , March 2004.

[60] Hao Jiang and Constantinos Dovrolis. Why is the Internet Traffic Bursty in Short Time Scales? In *SIGMETRICS '05: Proceedings of the 2005 ACM SIGMETRICS international conference on Measurement and modeling of computer systems*, pages 241–252, New York, NY, USA, 2005. ACM Press.

[61] Janey C. Hoe. Improving the Start-up Behavior of a Congestion Control Scheme for TCP. In *SIGCOMM '96: Conference proceedings on Applications, technologies,*

architectures, and protocols for computer communications, pages 270–280, New York, NY, USA, 1996. ACM.

[62] Ningning Hu, Li (Erran) Li, Zhuoqing Morley Mao, Peter Steenkiste, and Jia Wang. Locating Internet Bottlenecks: Algorithms, Measurements, and Implications. In *SIGCOMM '04: Proceedings of the 2004 conference on Applications, technologies, architectures, and protocols for computer communications*, pages 41–54, New York, NY, USA, 2004. ACM Press.

[63] Ningning Hu and Peter Steenkiste. Evaluation and Characterization of Available Bandwidth Probing Techniques. *IEEE Journal on Selected Areas in Communications*, 21(6), August 2003.

[64] H.Y. Wei and Y.D. Lin. A Survey and Measurement-Based Comparison of Bandwidth Management Techniques. IEEE Communications Society, Surveys & Tutorials, October 2003.

[65] J. Bartlett. Optimizing Multi-homed Connections. *Business Communications Review*, 32(1):22–27, January 2002.

[66] J. Case and M. Fedor and M. Schoffstall and J. Davin. A Simple Network Management Protocol (SNMP). In *RFC 1157*, May 1990.

[67] J. Hawkinson and T. Bates. Guidelines for Creation, Selection, and Registration of an Autonomous System (AS). In *RFC 1930*, March 1996.

[68] J. Martin and A. Nilsson and I. Rhee. Delay-based Congestion Avoidance for TCP. *IEEE/ACM Trans. Netw.*, 11(3):356–369, 2003.

[69] V. Jacobson. Pathchar: A tool to Infer Characteristics of Internet Paths. ftp://ftp.ee.lbl.gov/pathchar/, April 1997.

[70] Manish Jain and Constantinos Dovrolis. End-to-End Available Bandwidth: Measurement Methodology, Dynamics, and Relation with TCP Throughput. *IEEE/ACM Trans. Netw.*, 11(4):537–549, 2003.

[71] Manish Jain and Constantinos Dovrolis. End-to-End Estimation of the Available Bandwidth Variation Range. *SIGMETRICS Perform. Eval. Rev.*, 33(1):265–276, 2005.

[72] J.Moy. OSPF. RFC 2328, April 1998.

[73] K. Varadhan. BGP OSPF Interaction. RFC 1403, Jan 1993.

[74] S. Karandikar, S. Kalyanaraman, P. Bagal, and B. Packer. TCP Rate Control. In *SIGCOMM Comput. Commun. Rev.*, volume 30, number = 1, pages 45–58, 2000.

[75] A. R. Karlin, M. S. Manasse, L. Rudolph, and D. D. Sleator. Competitive Snoopy Caching. *Algorithmica*, 3(1):79–119, March 1998.

[76] R. Karp, E. Koutsoupias, C. Papadimitriou, and S. Shenker. Optimization Problems in Congestion Control. In *FOCS '00: Proceedings of the 41st Annual Symposium on Foundations of Computer Science*, page 66, Washington, DC, USA, 2000. IEEE Computer Society.

[77] Kin-Hon Ho and Ning Wang and P. Trimintzios and G. Pavlou and M. Howarth. On Egress Router Selection for Inter-domain Traffic with Bandwidth Guarantees. In *High Performance Switching and Routing, 2004. HPSR. 2004 Workshop on*, 2004.

[78] J.P. Knight. Review of Bandwidth Management Technologies, Availability, and Relevance to UK Education. "http://www.bmas.ja.net/papers/review/BMAS_Bandwidth_Management_Review.htm", December 2003.

[79] A. Kumar, M. Hedge, S.V.R. Anand, B.N.Bindu, D. Thirumurthy, and Arzad A. Kherani IISC Bangalore. Non-intrusive TCP Connection Admission Control for Bandwidth Management of an Internet Access Link. In *IEEE Communication Magazine*, May 2000.

[80] C. Labovitz, A. Ahuja, A. Bose, and F. Jahanian. Delayed Internet Routing Convergence. *IEEE/ACM Trans. Netw.*, 9(3):293–306, 2001.

[81] Will E. Leland, Murad S. Taqqu, Walter Willinger, and Daniel V. Wilson. On the Self-similar Nature of Ethernet Traffic (extended version). *IEEE/ACM Trans. Netw.*, 2(1):1–15, 1994.

[82] Yong Liu and A.L.N. Reddy. Route Optimization Among a Group of Multihomed Stub Networks. *Global Telecommunications Conference, 2005. GLOBECOM '05. IEEE*, 2:5 pp.–, 28 Nov.-2 Dec. 2005.

[83] M. Caesar and J. Rexford. BGP Routing Policies in ISP Networks. *Network, IEEE*, Nov.-Dec. 2005.

[84] B. Melander, M. Bjorkman, and P. Gunningberg. A New End-to-End Probing and Analysis Method for Estimating Bandwidth Bottlenecks. *Global Telecommunications Conference, 2000. GLOBECOM '00. IEEE*, 1:415–420 vol.1, 2000.

[85] Edwin E. Mier, Vincent J. Battistelli, and Alan R. Miner. Bandwidth Managers: Going With The Flow. "http://www.webtorials.com/main/resource/papers/BCR/paper55.htm".

[86] A. Odlyzko. Internet Traffic Growth: Sources and Implications[online]. http://www.dtc.umn.edu/odlyzko/doc/itcom.internet.growth.pdf., 2003.

[87] V. Paxson. Growth Trends in Wide-Area TCP Connections. *Network, IEEE*, 8(4):8–17, Jul/Aug 1994.

[88] B. Quoitin and O. Bonaventure. A Survey of the Utilization of the BGP Community Attribute. Internet Engineering Task Force, Internet Draft (work in progress) draft-quoitin-bgp-comm-survey-00.txt, March 2002.

[89] R. J. Shapiro. The Internet's Capacity to Handle Fast-Rising Demand for Bandwidth[online]. US Internet Industry Association, "http://www.usiia.org/pubs/Demand.pdf", September 2007.

[90] R. Pang and M. Allman and M. Bennett and J. Lee and V. Paxson and B. Tierney. A First Look at Modern Enterprise Traffic. In *SIGCOMM/USENIX Internet Measurement Conference*, Oct. 2005.

[91] R. Prasad and C. Dovrolis and M. Murray and K. Claffy. Bandwidth Estimation: Metrics, Measurement Techniques, and Tools. In *IEEE Network*, pages 27–35, Nov.-Dec. 2003.

[92] R. T. Marler and J. S. Arora. Survey of Multi-objective Optimization Methods for Engineering. *Structural and Multidisciplinary Optimization*, 26(6):369–395, 2004.

[93] Renata Teixeira and Timothy G. Griffin and Mauricio G. C. Resende and Jennifer Rexford. TIE breaking: Tunable Interdomain Egress Selection. In *CoNEXT'05: Proceedings of the 2005 ACM conference on Emerging Network Experiment and Technology*, pages 93–104, New York, NY, USA, 2005. ACM Press.

[94] Vinay J. Ribeiro, Rudolf H. Riedi, and Richard G. Baraniuk. Spatio-Temporal Available Bandwidth Estimation with STAB. *SIGMETRICS Perform. Eval. Rev.*, 32(1):394–395, 2004.

[95] Dan Rubenstein, Jim Kurose, and Don Towsley. Detecting Shared Congestion of Flows via End-to-End Measurement. In *SIGMETRICS '00: Proceedings of the 2000 ACM SIGMETRICS international conference on Measurement and modeling of computer systems*, pages 145–155, New York, NY, USA, 2000. ACM.

[96] S. Agarwal and A. Nucci and S. Bhattacharyya. Measuring the Shared Fate of IGP Engineering and Interdomain Traffic. In *ICNP '05: Proceedings of the 13TH IEEE International Conference on Network Protocols (ICNP'05)*, pages 236–245, Washington, DC, USA, 2005. IEEE Computer Society.

[97] S. Blake and D. Black and M. Carlson and E. Davies and Z. Wang and W. Weiss. An Architecture for Differentiated Services. In *RFC 2475*, December 1998.

[98] S. Sangli and D. Tappan. BGP Extended Communities Attribute. RFC 4360, February 2006.

[99] S. Uhlig and O. Bonaventure. Implications of Interdomain Traffic Characteristics on Traffic Engineering. *European Transactions on Telecommunications*, Jan 2002.

[100] Saad Biaz and Nitin H. Vaidya. Is the Round-Trip Time Correlated with the number of Packets in flight? In *IMC '03: Proceedings of the 3rd ACM SIGCOMM conference on Internet measurement*, pages 273–278, New York, NY, USA, 2003. ACM Press.

[101] J. H. Saltzer, D. P. Reed, and D. D. Clark. End-to-end Arguments in System Design. *ACM Trans. Comput. Syst.*, 2(4):277–288, 1984.

[102] S. Savage, A. Collins, E. Hoffman, J. Snell, and T. Anderson. The End-to-End Effects of Internet Path Selection. *SIGCOMM Comput. Commun. Rev.*, 29(4):289–299, 1999.

[103] Daniel D. Sleator and Robert E. Tarjan. Amortized Efficiency of List Update and Paging Rules. *Commun. ACM*, 28(2):202–208, 1985.

[104] L. Subramanian, S. Agarwal, J. Rexford, and R. H. Katz. Characterizing the Internet Hierarchy from Multiple Vantage Points. *INFOCOM 2002. Twenty-First Annual Joint Conference of the IEEE Computer and Communications Societies. Proceedings. IEEE*, 2:618–627 vol.2, 2002.

[105] T. Bates and Y. Rekter. Scalable Support for Multi-homed Multi-provider Connectivity. In *RFC 2260*, January 1998.

[106] T. Bressoud and R. Rastogi. Optimal Configuration for BGP Route Selection. INFOCOM 2003, April 2003.

[107] Tat Wing Chim and K. L. Yeung. Traffic Distribution over Equal-cost Multi-paths. In *IEEE International Conference on Communications*, volume 2, pages 1207–1211, June 2004.

[108] K. Thompson, G. J. Miller, and R. Wilder. Wide-area Internet Traffic Patterns and Characteristics. *Network, IEEE*, 11(6):10–23, Nov/Dec 1997.

[109] S. Uhlig and O. Bonaventure. Designing BGP-based Outbound Traffic Engineering Techniques for Stub ASes. *SIGCOMM Comput. Commun. Rev.*, 34(5):89–106, 2004.

[110] C. Williamson. Internet Traffic Measurement. *Internet Computing, IEEE*, 5(6):70–74, Nov/Dec 2001.

[111] X. Xiao, A. Hannan, B. Bailey, and L. Ni. Traffic Engineering with MPLS in the Internet. *IEEE Network Magazine*, 14(2):28–33, March/April 2000.

[112] Y. Rekhter and T. Li. and S. Hares. A Border Gateway Protocol 4(BGP-4). RFC 4271, Jan 2006.

[113] H. Zhou, Y. Wang, and Q. Wang. Measuring Internet Bottlenecks: Location, Capacity, and Available Bandwidth. *Networking and Mobile Computing*, 3619/2005:1052–1062, 2005.

[114] H. Zhou, Y. Wang, X. Wang, and X. Huai. Difficulties in Estimating Available Bandwidth. *Communications, 2006. ICC '06. IEEE International Conference on*, 2:704–709, June 2006.

[115] R. Moskowitz and P. Nikander, Host Identity Protocol (HIP) Architecture, RFC 4423, May 2006

[116] HIP Official charter, http://www.ietf.org/html.charters/hip-charter.html

[117] R. Moskowitz, P. Nikander, P. Jokela and T. Henderson, Host Identity Protocol, RFC 5201, April 2008