Load Balancing Inbound Traffic in Multihomed Stub Autonomous Systems

Ashok Singh Sairam and Gautam Barua
Department of Computer Science and Engineering
Indian Institute of Technology Guwahati
Guwahati 781039, India
Email: {ashok,gb}@iitg.ernet.in

***Abstract*—In the Internet, with many competing networks each trying to optimise its own bandwidth, a stub network has limited knowledge about user demands, available network resources and routing policies of other networks. This uncertainty makes the task of interdomain traffic engineering for a stub network very challenging. The basic aim of a stub network connected to multiple ISPs (multihomed) is to load balance its traffic among its various edge links. Our goal in this work is to distribute the incoming traffic of a multihomed stub network among its various edge links. The focus is on networks that primarily download traffic from the Internet. Regulating the incoming traffic is difficult since it will require to influence the behaviour of the remote destinations. We performed a systematic analysis of our problem and showed that even a restricted instance of the problem is NP-complete. We proposed simple, low-cost route control techniques to load balance traffic by reallocating the routes of outgoing traffic. The techniques were validated using synthetic as well as actual data collected under numerous traffic load conditions. Results show that we can achieve significant improvement in load balancing with minimum traffic re-assignments. Moreover, the proposed techniques neither require any third party assistance nor changes to existing protocols and network setup. This makes our schemes easily deployable in real networks.**

I. Introduction

One of the driving forces behind the evolution of Internet is due to a substantial growth in the number of stub autonomous systems (ASes). Transit ASes primarily interconnect networks and hence they are connected to more than one provider or peer. Interestingly this trend of multihoming has been observed in stub ASes too ([26]). In multihomed networks, traffic on one path is congested, even when traffic on other links are under-utilised. Re-balancing the traffic load can result in significant improvements in download speeds. A number of tools and techniques collectively referred to as *smart routing* or *intelligent route control* have been proposed for exploiting multihoming to extract performance benefits ([7],[18]). However, these tools concentrate on improving the end-to-end performance, rather than redistributing the traffic. The performance gains of our multihoming route control technique are comparable to existing multihomed techniques yet our approach is much simpler and practical. To realise performance benefits, existing multihomed techniques rely on choosing the best ISP link and assume that the stub network has perfect information about the performance of the ISPs. ([7]). In this work we realise similar performance benefits by optimising the utilisation of the multiple edge links at our disposal.

In the Internet, a stub network has limited knowledge about user demands, available networks resources and routing policies of other networks. Tools available to measure network metrics do not give accurate results. Moreover Internet traffic is highly chaotic when considered at small time scales. The presence of such uncertainties makes the task of load balancing in an interdomain environment very challenging. For wider acceptability, the need to propose solutions within the framework of existing protocols and standards makes the problem even more difficult. As the size of the network grows, intradomain path between a user and an edge router can be several hops. Consequently, intradomain traffic can have a substantial affect in the overall latency ([22]). Thus while selecting the edge links we will also need to consider the intradomain bandwidth.

The main goal of this paper is to distribute the incoming traffic of a multihomed stub network among its various edge links. We tackle the inaccuracies of network monitoring tools, by attempting to keep link utilisations proportionate to their available bandwidth. Such a step will ensure that the link utilisations are fairly balanced even if the measurements of available bandwidth is an approximation. The other two goals are to minimize the re-assignments and to incorporate the intradomain cost in the route selecting process. We attempt to solve the problem in two steps. In the first phase, we assume that the input traffic is known and that the bandwidth measurements are accurate. Even with such strong and unrealistic assumptions, we find that the problem is NP-complete. This part of our study is called the *offline analysis*. In the second phase, we assume the input traffic is completely unknown. We call this part of our study the *online analysis* or *online models*. Leveraging upon our approaches developed for the offline case, we propose route control mechanisms for the *online models*. The proposed techniques although simple yield significant improvement in traffic load balancing. Empirical results show that our techniques yield more than 40 percent improvements as compared to a scenario where no load balancing techniques are used.

The rest of the paper is organized as follows. A brief background to our problem is presented in section II. In section III we describe the problem. The offline and online models are discussed in section IV and V respectively. The experimental

results are given in section VI. Finally the concluding remarks and future work are presented in section VII.

II. Background and Motivation

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 ([25]). 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. The traffic in access networks will mainly consist of small outgoing requests and large response. The main objective of this paper is to load balance the traffic that enters a network.

BGP ([27]), the current de-facto interdomain routing protocol has not been primarily designed from a perspective of interdomain traffic engineering. Nevertheless, a number of tools have been designed to improve Internet performance by tweaking BGP policies. These tools have been primarily designed for outbound traffic and hence they are not competent to handle incoming traffic ([26], [21]). Edge based traffic engineering techniques([23]), which identify and categorize specific network traffic, and then constrain each category to use no more than a specified amount of bandwidth will not be effective on inbound traffic since they act on the traffic after it has already consumed the scarce bandwidth. Inbound 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, a stub AS monitors the incoming bandwidth on all its edge links. When the stub network observes that a particular edge 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 edge link. As the focus of our work is on access networks, requests for downloads will primarily originate from within the network. Besides, Internet traffic being primarily in the form of request-response pairs, re-routing can be achieved by scheduling the outgoing request appropriately.

III. Problem Description

A. Measuring Available Bandwidth

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. To optimise the Internet experience of end users, it is crucial to understand the location and traffic load of bottleneck links. A stub network may have a dedicated link between itself and its uplink provider, but beyond a certain point its Internet traffic flows will share network resources with millions of other flows. The stub network does not have any control on the network traffic beyond a point. The aim of this work is to optimise the utilisation of resources at the disposal of the network. We assume that there is a point "X" in the provider's network to which all packets from the network have to travel to, and that beyond the point X, there is sufficient bandwidth available. The congestion (or restriction in available bandwidth) is assumed to be only up to point X. The path to this point X from the edge 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. We adapt an existing tool pathneck ([19]) to infer the congestion point and measure the available bandwidth of the *connecting path*.

B. User Traffic

The only assumption made about the traffic is that we can split the traffic entering the network at a certain level of granularity. Other than this we do not make any assumptions about the characteristics of the 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 default gateway of all the users will be cumbersome and unmanageable. To keep the routing tables simple and manageable, we aggregate the users into user classes. In this work the quality of service(QoS) we provide is in terms of user movements. The classes can be formed based on similar QoS requirements of users, and a priority can be assigned to each class based on the QoS needs. We select user classes with lower priorities as candidates for re-assignment. This will ensure that only low priority users will face possible disruptions due to movement.

C. Network Model

Assuming that all intradomain and edge links connecting to other ASes are bi-directional, the only requirement for our scheme to work is that the request and response of a traffic flow must follow the same edge route. That is if a request to download a file is send through an edge link L1 (say) then the corresponding response should also return through L1. We describe two network models that can support this assumption - NAT-based model and BGP-based model.

In order to preserve the limited number of IP addresses, network address translation (NAT) is often used to send outbound packets, with a source IP address associated by the ISP with an edge link. The corresponding return traffic will automatically come back via the same link, because it is the only link servicing that address range. NAT-based solutions are by far the most practical solution for controlling incoming traffic and they have been frequently used to control the flow of incoming traffic ([7], [18], [8]). 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 solution fails to scale, BGP route advertisements are manipulated to influence the incoming traffic ([13], [9]).

Selective advertisement is a technique commonly used in BGP to control incoming traffic. In this work we propose a variant of selective advertisement for our network model. Let us consider a stub Autonomous System, AS1 (as shown in figure 1), with two edge links e_1 and $.2$ and an independent IP address block. We partition the IP address prefixes into groups and then advertise these prefixes. Once we change the default gateway of a user group, say from link e_1 to e_2, we announce the prefix of the user group on link e_2 and withdraw it from e_1. The advantage of this model is that as soon as 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. An issue with the use of sub-prefixes is that many ISPs filter out small IP prefixes advertisements. 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 ([9]). The other issue is with regard to time taken for route convergence. BGP route convergence typically range from seconds to tens of minutes ([20]). 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 ([10]). The heuristics that we propose proceeds in periods. During each period it is expected that there will be some route updates. However, 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 ([15]) which allow network operators to predict flow of traffic due to changes in BGP policies.

The periodic link state updates of intradomain protocols do not include any traffic engineering metrics. To facilitate intradomain traffic engineering, extensions to link state protocols have been proposed. OSPF-TE ([14]) 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.

D. Problem Statement

Consider a network with N edge links, $\mathrm{E} = \{\mathrm{e}_1, \mathrm{e}_2, .., \mathrm{e}_\mathrm{N}\}$. Let S be the size of the user population and the set of users be, $\mathrm{I} = \{\mathrm{i}_1, \mathrm{i}_2, ..\mathrm{i}_\mathrm{S}\}$. Time is divided into discrete intervals, indexed by t=$\{1,2,...\}$. A period t denotes the time interval (t-1,t). The available bandwidth of a link *e* (in Mbps) is denoted by $\mathrm{A_e}$. Let $\mathrm{U_e(t)}$ (in Mbps) denote the measured incoming traffic of link *e* and let $\mathrm{b_i(t)}$ denote the incoming traffic of user *i* in the period t. The variable $\mathrm{x_{ie}(t)}$ has a value of 1 if user *i* is assigned link *e* at time t, otherwise it has a value 0. A user can be assigned to only one edge link at any instant. Therefore, $\sum_{e=1}^{N} x_{ie}(t) = 1,\ \forall i \in I$. The objectives that we attempt to achieve are:

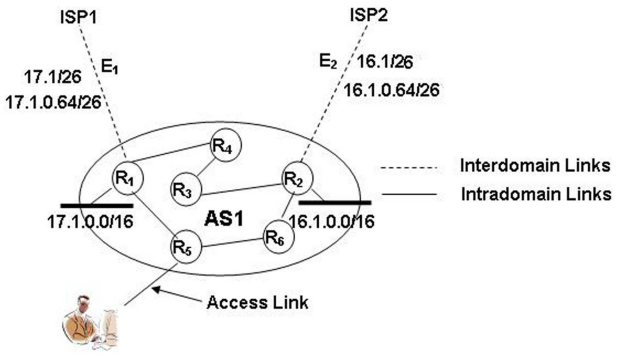


Fig. 1. Managing user movement in a stub AS.

- The utilisation of the edge links is in tune with their available bandwidth. This objective will ensure that the actual utilisations of the links are fairly balanced although our measure of link utilisation is approximate. We define the ideal utilisation of a link as:

$$IU_e(t) = K\ .\ A_e(t) \quad \text{where} \quad K = \frac{\sum_{l=1}^{N} U_l(t)}{\sum_{l=1}^{N} A_l(t)} \tag{1}$$

  Our objective is to keep the actual utilisation as close as possible to its optimal value.

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

- The number of user movements is as small as possible. That is we wish to minimize

$$f2:\ min \sum_{e=1}^{N} \sum_{i=1}^{S} |x_{ie}(t) - x_{ie}(t-1)| \tag{2}$$

  We say that a user has moved if its present assignment is different from its previous assignment. For each user movement, the value of the above equation will increase by 2.
- In larger networks like that of a stub AS, intradomain traffic can also have substantial affect on the overall latency. The intradomain path between a user and an edge router may be several hops. Ideally we would like each user to select an edge router that is the least costly to reach. If d(i,e) denotes the intradomain cost of the path between the user *i* and edge router *e*, then

$$f3:\ d(i,e) \leq d(i,e'),\ \forall e' \in E \tag{3}$$

IV. Offline Models

A major impediment 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

input traffic, though some of these assumptions may sound unrealistic. For each such assumption we examine the possible solution and their complexity.

1) Input Traffic is Known: Let us assume that the incoming traffic of users, that is value of the $b_i(t)'s$, are known at the time of assigning the users to links. The sum of these $b_i(t)'s$ will yield the total incoming traffic. Further we assume that the available bandwidth of each link is a constant equal to the link capacity. Since we know the assignment of users to links, we can also compute the utilisation of each individual link. Using equation 1, we compute the ideal utilisation of each edge link. Our job now is to distribute the user traffic such that utilisation of each link is at its ideal value.

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 edge links and to list the over-utilised and under-utilised links is given in algorithm 1.

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**
  $T\_Abw \leftarrow T\_Abw + A[e]$
end
/* U[] Measured Incoming Traffic */
for i = *1* **to** S **do**
  $T\_traffic \leftarrow T\_traffic + b[i]$
  $U[x[i]] \leftarrow U[x[i]] + b[i]$
end
/* IU[] Ideal Utilization */
for e = *1* **to** N **do**
  $IU[e] \leftarrow \frac{T\_traffic\ .\ A[e]}{T\_Abw}$
  $rank[e] \leftarrow IU[e] - U[e]$
  if *rank[e]* < 0 **then**
    /* OUtil[] Over-utilised links */
    $OUtil[index++] \leftarrow$ e
  end
  else if *rank[e]* > 0 **then**
    /* UUtil[] Under-utilised links */
    $UUtil[index1++] \leftarrow$ e
  end
end

Algorithm 1: Compute rank of edge links

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 utilisation of the links become ideal. 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.

Informally, we show that the partition problem, a well-known NP-complete problem ([24]) can be mapped to the re-stricted re-assignment problem. Given a multi-set S of integers, the partition problem is to decide whether there is a way to partition S into two subsets S1 and S2 such that the sum of the numbers in S1 equals the sum of the numbers in S2. In the restricted re-assignment problem, let us restrict the number of edge links to two. There is a one-to-one correspondence between the partition problem and the restricted re-assignment problem. Let us equate the set S to the user set I of our problem. The restricted re-assignment problem is reduced to the problem of splitting the users in the set I into two subsets, such that the sum of the incoming traffic of the users (i.e. $b_i's$) in each set is equal. A user cannot be assigned to two edge links simultaneously, therefore the subsets form a partition in the sense that they are disjoint and cover I. Since the partition problem is NP-complete, so is the restricted version of our problem.

We next establish an interesting correlation between over-utilised and under-utilised links. If one or more of the edge links are over-utilised then there are guaranteed to be under-utilised link(s) which will exactly fit the excess traffic. A formal proof of this statement is given in appendix A (theorem 8.1).

2) Input traffic is Known and Bounded: As mentioned in section III-A, our main goal is to optimise the bandwidth of the *connecting path*. Incoming traffic of a user cannot exceed the available bandwidth of this path. We, therefore, define an upper bound on the user bandwidth equal to the available bandwidth of the *connecting path*. The problem is now equivalent to a partition problem where the "size" of the elements of the set is bounded. Garey and Johnson ([17]) show that in the partition problem, *pseudo-polynomial* time bounds solutions are possible if the "sizes" have a upper bound. Let us further restrict the values of the user bandwidth to two - 0 or 1. A value of 0 means the user is not receiving any traffic (dormant) while 1 means the user is receiving traffic at the maximum possible bandwidth (active). Then from theorem 8.1, 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. A straight forward solution 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 $\bigodot(S)$ time, where S is the size of the user population.

The conclusion that we draw from our discussion of the offline models are - (i) an exact solution to the problem cannot be realised even if the input traffic is assumed to be known,

and (ii) feasible solutions can be realised if the input traffic is suitably discretized.

V. Online Models

The offline analysis has been carried out to gain insight into the problem. In the real world input traffic will change dynamically and their values will be unknown at the time of assigning users to links. In this section we remove all restrictions imposed on the input traffic. We do not make any assumptions about the input traffic. Therefore, 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.

A. No Restrictions on Input traffic, Intradomain Traffic Static

In this section we assume the input traffic 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 edge links can vary.

Greedy Approach: Our first heuristic is to move the user that is currently receiving the maximum traffic. Studies have shown that a few high throughput users contribute majority of the traffic. Our aim is to identify these users and move them so as to achieve load balancing with minimum re-assignment. We call this heuristic the greedy approach. The heuristic proceeds in stages or periods. The duration of a period is set as five 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 ([25], [5]). At the end of each period the following actions are performed:

1) Compute the ranks of the links based on the current network traffic state. This is done using algorithm 1.
2) Sort users assigned to over-utilised links in descending order of their incoming traffic. Consider them one by one.
3) 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 link go to next step, else repeat 3 till a user can be moved; exit if no more users are left in the sorted list.
4) Move user, re-compute link utilisations and ranks.
5) Repeat steps 3 and 4 until the utilisation of all the links become optimal or no user can be moved.

We say that a user *fits* into a link if assigning the user to the link does not make the link over-utilised. As the users are moved the utilisation of the links are updated accordingly. The heuristic proceeds recursively with the second stage acting as the first for the next run of the algorithm. A pseudo-code of the greedy approach is given in algorithm 2. Depending on the manner we select users for re-assignment, there can be different algorithms. Another approach would be to select the users based on the outcome of a random game and *relative utilisation* of their edge links. In this paper we stick to the greedy approach only.

During the course of our experiments, we discovered that when a high throughput user is moved from an over-utilised link to an under-utilised link, the second link becomes over-utilised and the first one under-utilised. Thus in the next period the user gets moved back to the first link and in this way the user oscillates between links. To confront such user oscillations, we incorporated an additional constraint. 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-assignment 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 entire period of the simulation. The duration of our simulations range from 1 hour to 2 hours.

B. Input Traffic Unknown, Network State Dynamic

In this section, 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. Thus while selecting an under-utilised link, we additionally need to take into account the dynamics of intradomain traffic. In existing Internet protocols like BGP, if the route selection process results in multiple equally good edge links then the intradomain distance is used to break the tie. 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 edge links. Thus from this 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, that is the *getnext* function. A pseudo-code of the new *getnext* function is given below:

VI. Experimental Results

In this section we describe our experimental evaluation of the greedy approach. First, we evaluated the performance of the route control model proposed for static intradomain network environment and examined the sensitivity of the model to cross-traffic. In the second and third experiments we analyzed our dynamic route control algorithm using synthetic and actual traffic traces respectively. To quantify the benefits of our approach we compare our heuristics to the *default* case, when no load balancing mechanisms are applied.

A. Data Set

Simulation results will largely depend on the characteristics of the traffic traces. The traces should be collected at transit

```
input : OUtil[], UUtil[]
/* 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]
    /* Get most under-utilised link        */
    next ← getnext()
    /* Check if user ou fits.              */
    if (user ou fits link next) then
        /* Move user.                      */
        x[ou] ← next
        /* Update link utilisation and
           rank                            */
    end
end
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 2: Greedy Algorithm

```
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
```

points so that individual traffic from all the users can be seen. Secondly the traces should ideally cover traffic on at least two different interdomain links. 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. In this section we present our observations from the analysis of a 1 hour trace collected from 4 different ISP links. Network traffic traces were collected using tcpdump ([3]). 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 1 sec only. Traffic flows are usually classified on the basis of their size and lifetime. We analyzed the trace both in terms of the bytes downloaded and lifetime of a user. The data set exhibited all the characteristics of interdomain traffic. A summary of our analysis is given below.

- The users were sorted on the basis of their size. We found that the top 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 first bin considered users that had a lifetime of less than 5 minutes, the second bin considered users that had a lifetime ranging between 5 to 10 minutes and so on. Out of the 12 bins, 40 percent of the users fell in the first bin. However, the total bytes downloaded by these users were just 4.5 percent. That is majority of the users are short-lived and have a low throughput (mice). Such users will not have much of an impact on load balancing of the links.
- There does not seem to be any correlation between the size and lifetime of a user. Earlier studies ([11]) have also indicated that flow size is independent of the flow lifetime.

B. Simulating under Static Intradomain Traffic Conditions

In this experiment, we simulated the greedy algorithm using a simple simulation program developed using Perl ([2]). The program initially analyzes the traces individually and sums the payload of all incoming packets that fall within a period. This sum represents the *default* utilisation of the links, that is the utilisation when no load balancing techniques are used. While computing the utilisation of each link, we also list the distinct users present in the trace. These users represent the *default* assignment of users to the link. Subsequently, we merge the traces in chronological order using tcpslice ([4]). Now we simulate the greedy algorithm on this merged trace. The assignment of users is changed as demanded by the algorithm and the link utilisations are computed accordingly. This simulation program does not consider the intradomain traffic dynamics but nevertheless we can get a feel of the performance of the algorithms in actual traffic conditions.

1) Experiment 1: 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 edge links considered was four, the trace duration was of one hour. While the traces were collected, we also recorded the available bandwidth of the links. There were three different runs of the experiment. In the first run no load balancing techniques were used and we simply computed the utilisation of the links in each period (*default*). In the second run of the experiment, we used the greedy approach to re-assign the route of the users but no cross traffic were considered. In the third run of our experiment, we simulated cross traffic by varying available bandwidth of the links in each period as per our recorded values.

The rank of a link is a measure of the deviation of the utilisation of the link from its ideal value. The overall deviation for the *default* case, when no cross traffic was considered, was 18.85 and 17.1 when cross traffic was considered. To quantify the improvement in load balancing achieved, we computed period-wise the difference in deviation of the utilisation of the links when the greedy approach was used to that of the *default* approach. When the greedy approach was used, the deviation of the links from their ideal value was significantly less. The overall improvement in link deviations is shown in table I. The improvement in load balancing was 53 percent when cross-traffic was not considered and 43 percent when cross traffic was considered. Moreover, we find that the mean and median are almost same. This indicates that the performance improvement is uniform in all the periods. The mean, median and standard deviation of the re-assignments are given in table II. The number of re-assignments is also identical in all the periods. This means that due to the dynamic nature of Internet traffic we constantly need to load balance the traffic.

TABLE I
IMPROVEMENT OF ROUTE CONTROL SCHEME IN COMPARISON TO *default* CASE (IN PERCENT)

Mean	Median	Standard Deviation
No Cross Traffic		
9.96	9.19	3.78
Cross Traffic		
7.34	7.09	5.95

TABLE II
USER RE-ASSIGNMENTS (IN PERCENT)

Mean	Median	Standard Deviation
No Cross Traffic		
1.76	1.3	1.83
Cross Traffic		
0.65	0.3	1.0

C. Simulating under Dynamic Traffic Conditions

In order to incorporate the dynamics of intradomain traffic into our simulation, we first need to create a topology of the internal network. As far as choosing a network topology was concerned we used a standard topology generator (BRITE ([6]). The network model was later enhanced by dynamically changing the link attributes, bandwidth and link costs. To find how various attributes of network topologies 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 edge nodes and access nodes. We assumed that an edge router connects to a single ISP. To simulate the edge links between the stub network and its ISPs (*connecting path*), for each edge router 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 that of the intradomain links were changed. Cost of a link was set to the inverse of its available 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 ([1]).

User nodes were created and attached to the access nodes. For each user node we created a corresponding destination node (sink) and attached it to its relevant ISP node. The cost of the link from the user node to the access node and from the destination node to the ISP node was set to 1. To simulate our proposed route control techniques, we dynamically need to change the route assignment of users. In order to effect such 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 2 shows a network topology with two edge routers, two ISPs and three user classes. As can be seen in the figure, USER1 and USER2 are assigned to ISP1 and USER3 is assigned to ISP2. For instance, if during the simulation we require to move USER1 from ISP1 to ISP2, then the link from DST1 to ISP1 is set to infinity and the link from DST1 to ISP2 is set to 1.

1) Experiment 2: Validating Using Synthetic Data: A major challenge while developing test beds for network experiments is modeling the Internet traffic. The discrete event simulator, ns-2, provides a rich library of traffic models. In this experiment 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 ([12]), a model for generating HTTP traffic. The second user type used was the ns-2 class *PagePool/WebTraf*, a standalone web traffic model ([16]). The third user class was a ftp server. Internal or intradomain traffic was generated using *Pareto On/Off* traffic sources. The topology considered consisted of 25 nodes and 50 intradomain links.

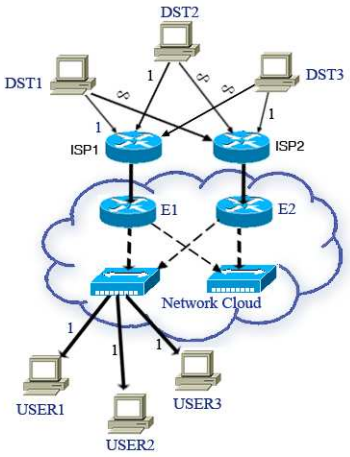


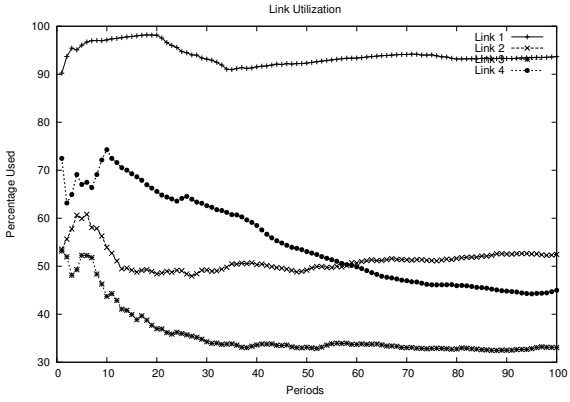
Fig. 2. Network topology.

The number of edge routers considered was four. Capacity of all the links were considered to be the same (10 Mbps). The duration of the experiment was 100 periods. 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 proposed for a dynamic network environment with the same network and user class settings. To ensure that the same amount of traffic was generated in both the runs, the user class parameters were kept same. The link utilisations when no load balancing techniques were used and when the greedy approach was used are shown in figure 3(a) and 3(b) respectively. As can be seen, application of our route control strategy significantly improved traffic load balancing on the links. The average deviation of the links from their ideal value (i.e. ranks) was reduced from 30.67 percent to 6.03 percent. The load balancing of the links have improved by 80 percent. The overall user re-assignment was about 4.48 percent. The relatively higher percentage of user re-assignment is because the *default* deviation of the links was very high. One goal of this experiment was to test the extent of performance improvement in load balancing possible on links with highly disparate utilisations. In real network conditions, we find that deviations for the *default* case are relatively less and hence the user re-assignments are also low. An advantage of using synthetic data is that we can store the request-response exchanges between a client and a server. In table III, the outgoing traffic, incoming traffic and round-trip time (RTT) experienced by the users is shown for both runs of the experiment. The round-trip times improved by about 7 percent as compared to the *default* case. The volume of outgoing and incoming traffic in both runs of the experiment was almost the same. This implies that traffic condition in both runs of the experiment was similar.

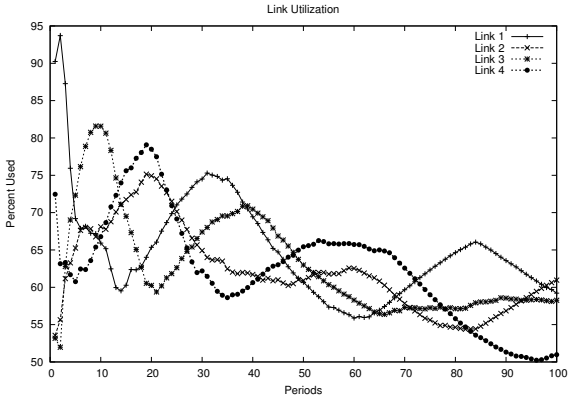
2) Experiment 3: Validating Using Actual Data Traces: In this experiment we propose to use the actual data, but unfortunately ns-2 does not directly support TCP connection

TABLE III
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



(a) Default



(b) Greedy

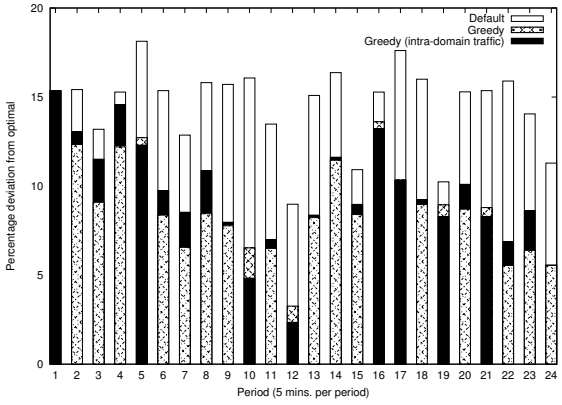
Fig. 3. Expt 2: Plot of link utilisations.

from traffic traces. The individual traces collected from each of the edge links were therefore pre-processed before being fed into ns-2. We first extracted the distinct users (IP addresses) present in a trace. Individual ns2 trace files were created for each of these users. The trace files were scanned and whenever 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. Like in our previous experiment for each user node we created a corresponding destination node. But instead of attaching a traffic model to the user node, the ns2 trace of the user was streamed from the destination to the user.

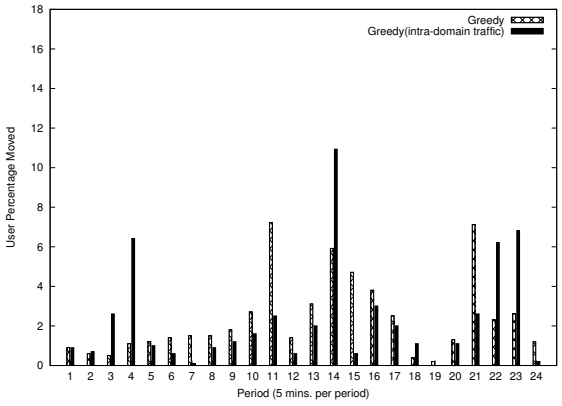
In this experiment, the user base considered was 1500 users. The duration of the trace was 2 hours. The total traffic downloaded was about 20 gigabytes and the outgoing traffic

was 1.36 gigabytes. The network topology consisted of 50 nodes and 300 intradomain links. The numbers of edge and access nodes considered were 8 and 17 respectively. With these experimental settings, we simulated the three route control mechanisms - *default* case, the greedy algorithm that does not consider intradomain traffic and the greedy algorithm that considers intradomain traffic while selecting routes. The deviations of the links from their ideal value for all the three schemes were plotted using a stacked histogram (figure 4(a)). A stacked histogram for each period allows comparison in the following way. The first stack depicts the least deviation, the second stack shows the next higher deviation. The third stack shows the highest deviation. As can be seen in the figure, 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. Both the greedy approach performed better than the *default* case in all the time periods. The average percentage deviation of the link utilisations from their 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). Thus, the performance improvement in terms of load balancing as compared to the *default* case is on an average about 40 percent. The user re-assignments are shown in figure 4(b). The overall percentage of users re-assigned per hour is less than 2 percent.

In an actual network, latency of 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 intradomain paths followed by a user, the delays incurred by the incoming traffic will also differ. However, in this simulation, we cannot see the effect of such delays as our incoming traffic is pumped at a pre-defined rate. Nevertheless, to bring to light the difference, 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 edge links were selected without considering intradomain traffic was 1295.91. In contrast if the routes were selected based on the IGP cost, the total cost of the intradomain paths was 590.65. Thus there is an overall improvement by more than 50 percent in terms of the IGP cost. In real network this improvement in the intradomain cost will improve the RTTs of the downloads.



(a) Comparison of Ranks



(b) Percentage of Users Re-assigned

Fig. 4. Expt 3: Greedy approach without and with intradomain traffic.

VII. Conclusion

A number of previous works ([7], [8]) have established the practical benefits of load balancing traffic. In this work we propose to load balance the incoming traffic of a stub access network by keeping the link utilisations in proportion to their available bandwidth. The scheme ensures that utilisations of the links are fairly balanced even if the measurement of bandwidth metrics is approximate. In the first part of our work we assume that the network traffic is known and the measurements are accurate (offline models). Even with such strong assumptions we prove that our problem is NP-complete. We also prove that if there are *over-utilised* links then their must exist *under-utilised* links that will exactly accommodate the excess traffic. In the later part of our work, no assumptions were made about the input traffic (online models). Leveraging on our offline analysis, we propose a simple, light-weight and practically viable solution (greedy approach). Our proposed scheme can be deployed with minimal changes to existing networks. The proposal was tested under different topologies and network traffic loads. First we tested the greedy heuristic in a static intradomain traffic environment and examined the effects of cross traffic. Next we examined the performance in a dynamic network environment using synthetic data. Finally the performance was tested using real traffic traces. Results show that we can achieve more than 50 percent improvement in load balancing traffic in a static intradomain network environment and 40 percent improvement in a dynamic network environment. The re-assignment cost was small, less than 2

percent of the total users present. There is, however, a need to carry out more experiments in a real network and to study the effects of our route control techniques on neighbouring domains. As future work we intend to develop a distributed version of our algorithm.

VIII. Appendix A

Theorem 8.1: 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 $A_e(t)$ denote the ideal utilisation, actual utilisation and available bandwidth of a link of a line *e* at time *t*. Our basic goal is to re-distribute the total incoming traffic. Therefore, the sum of ideal utilisation of the links is equal to the sum of the utilisation of the links.

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

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

$$(IU_1 - U_1) + (IU_2 - U_2) + \ldots + (IU_N - U_N) = 0 \quad (4)$$

Each term in equation 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 the utilization of all the links are ideal. 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 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. ■

References

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

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

[3] tcpdump - dump traffic on a network. http://www.tcpdump.org/.

[4] tcptrace. http://www.tcptrace.org/.

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

[6] 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.

[7] 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.

[8] A.S. Sairam and G. Barua. Effective Bandwidth Utilisation in Multihoming Networks. *Communication System Software and Middleware, 2006. Comsware 2006. First International Conference on*, pages 1–8, Jan 2006.

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

[10] Olivier Bonaventure, Clarence Filsfils, and Pierre Francois. Achieving Sub-50 Milliseconds Recovery upon BGP Peering Link Failures. *IEEE/ACM Trans. Netw.*, 15(5):1123–1135, 2007.

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

[12] 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.

[13] 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.

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

[15] Nick Feamster, Jay Borkenhagen, and Jennifer Rexford. Guidelines for Interdomain Traffic Engineering. *SIGCOMM Comput. Commun. Rev.*, 33(5):19–30, 2003.

[16] 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.

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

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

[19] 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.

[20] Craig Labovitz, Abha Ahuja, Abhijit Bose, and Farnam Jahanian. Delayed Internet Routing Convergence. *IEEE/ACM Trans. Netw.*, 9(3):293–306, 2001.

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

[22] 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.

[23] 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.

[24] S. Martello and P. Toth. *Knapsack Problems*. John Wiley & Sons, 1990.

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

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

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