An approach to alleviate link overload as observed on an IP backbone

Sundar Iyer[1*], Supratik Bhattacharyya[2], Nina Taft[2], Christophe Diot[2]

[1]Computer Systems Laboratory,
Stanford University,
Stanford, CA 94305-9030
sundaes@stanford.edu

[2]ATL Sprintlabs,
1 Adrian Ct,
Burlingame, CA 94010
{supratik, nina, cdiot}@sprintlabs.com

Abstract -- ***Shortest path routing protocols may suffer from congestion due to the use of a single shortest path between a source and a destination. The goal of our work is to first understand how links become overloaded in an IP backbone, and then to explore if the routing protocol, — either in its existing form, or in some enhanced form could be made to respond immediately to overload and reduce the likelihood of its occurrence. Our method is to use extensive measurements of Sprint's backbone network, measuring 138 links between September 2000 and June 2001. We find that since the backbone is designed to be overprovisioned, link overload is rare, and when it occurs, 80% of the time it is caused due to link failures. Furthermore, we find that when a link is overloaded, few (if any) other links in the network are also overloaded. This suggests that deflecting packets to less utilized alternate paths could be an effective method for tackling overload. We analytically derive the condition that a network, which has multiple equal length shortest paths between every pair of nodes (as is common in the highly meshed backbone networks) can provide for loop-free deflection paths if all the link weights are within a ratio*** $1 + 1/(d-1)$ ***of each other; where*** d ***is the diameter of the network. Based on our measurements, the nature of the backbone topology and the careful use of link weights, we propose a deflection routing algorithm to tackle link overload where each node makes local decisions. Simulations suggest that this can be a simple and efficient way to overcome link overload, without requiring any changes to the routing protocol.***

***Keywords*--Deflection Routing, Network Measurements, Link Overload, Link Failure, Traffic Engineering.**

I. INTRODUCTION

IP backbones are engineered for high availability and resilience to multiple link and router failures. It is therefore common for there to be multiple disjoint links, each of which have sufficient capacity to individually sustain the offered load, between any two neighboring Points of Presence (PoPs).

In addition to redundancy, the presence of multiple high capacity links has two consequences. First, it means that IP backbones in practice are designed to guarantee a path between any two points, even if multiple (but limited) number of links fail simultaneously. Second, it means that even when some links are overloaded, it is likely that other links are not, making load-balancing across the multiple available paths appealing.

The two widely used interior gateway routing protocols IS-IS[1] and OSPF[2] perform shortest path routing. The limitations of shortest path routing are well known. They constrain the load on the network along a small set of shortest paths even if alternate paths with available capacity exist and limit the throughput between any two nodes to well below the maximum capacity [1]. Ideally, if multiple paths were available, the routing protocol would spread the traffic uniformly over the whole network, minimizing the probability that any one link is overloaded. Indeed, both IS-IS and OSPF have been extended to split traffic among multiple equal cost paths. Multipath routing protocols which can divide the load unequally amongst several alternate paths have been suggested in literature [2][3][4].

Although intuition suggests that in an overprovisioned network, load balancing across multiple paths will reduce the likelihood of link overload, we are not aware of any published research that measures and shows this to be effective in an IP backbone, or indicates that there is still room for improvement in these protocols. This is in part due to the non availability of widespread measurement infrastructure. Unfortunately, when measurements are obtained, the data is usually of a proprietary nature.

The goal of our work is to first study Sprint's IP backbone — which uses IS-IS with load balancing across Equal Cost Multipaths (ECMP [5]) — and to understand how overload occurs in the backbone. We then explore whether the routing protocol needs to be enhanced to respond immediately and alleviate overload.

To this end, we study data collected over a nine-month period (September 2000 to June 2001) from 138 links on Sprint's IP backbone. Our measurements include the utilization of each link, the configuration of each router, and the IS-IS link weights.

We find that there is a wide-disparity in the load levels of different links. We observe that though most links in the network are not highly loaded, at any given time there is always some link which is overloaded. (In the rest of the paper, we will define overload to mean a link utilization above 50%[3]). Another significant observation is the occurrence of sudden and aperiodic "hotspots", where the link utilization is as high as

1. Intermediate System to Intermediate System, IS-IS, http://www.ietf.org/html.charters/isis-charter.html

* The author is currently affiliated with Stanford University, but this work was done during an internship with Sprintlabs.

2. Open Shortest Path Routing (OSPF), http://www.ietf.org/html.charters/ospf-charter.html

3. The backbone is engineered with the goal of keeping the load level of every link under 50% in the absence of link failure. This is done to ensure that when a link fails, there is enough capacity on the backup path to absorb all the traffic that fails over to it.

90%. We find that this is mostly caused due to link failures. However, at any given time, there is usually only one such "hotspot" in the network.

These observations suggest that the current approach to backbone design based on over-protection appears sufficient in most cases to keep network links from being overloaded. However, the main problems of link overload arise in the case of short term link failures (which last for not more than a few minutes), and which contribute to 80% of observed overload. Hence a solution is required, which can alleviate short term overload much faster than what network operators can do by manually changing link weights.

Based on these measurements on link utilization, the occurrence of overload, and the lack of simultaneous overload in the backbone, we believe that a simple deflection routing technique, where each router makes local decisions and allows a router to divert traffic from the default outgoing interface (determined by IS-IS) to an alternate interface that is less loaded, can be quite effective in tackling overload.

However, it is important that deflection routing not create routing loops in the network. So we begin by analytically deriving the conditions on a network, such that a distributed deflection routing algorithm is loop-free. We show that a network which has many equal length paths between any source and destination (like the Sprint backbone) can provide for loop-free deflection paths if all the link weights are within a ratio $1 + 1/(d-1)$ of each other; where d is the diameter of the network. Further we derive a less stringent condition on the link weights, which also ensures that the deflection routing algorithm is loop-free. We observe that this condition is almost always satisfied in the Sprint network. These observations lead us to suggest a practical deflection routing algorithm.

Note that deflection routing has already been proposed previously for both telephone [6] and data networks [7], as a mechanism to overcome overload. Our technique differs from previous work in judicious use of link weights to achieve a distributed loop-free deflection routing algorithm. We compare our proposal to previous work in more detail in Section II.D.

Our paper is organized in two parts. In the first part, we perform measurements on the backbone and lay the case for deflection routing. Specifically, Section II.A explains the methodology of evaluation, Section II.B describes our observations on link utilization, and Section II.C describes the occurrence of overload. We discuss why we believe deflection routing can be effective in tackling overload in Section II.D.

The second part of the paper analytically derives the requirements on the design of a loop-free deflection routing algorithm (Section III.A and B) and then describes the design of a practical deflection routing algorithm (Section III.D). Finally, we describe a number of simulations based on the Sprint topology that suggest that deflection routing can be effective in Section IV.

II. The Nature of Link Utilization and Overload on the Backbone

We first describe the sources from where we collect data, followed by a detailed description of our approach.

A. Methodology

1) Measurement Infrastructure: We collect three distinct types of data from the Sprint IP backbone. They are —

SNMP Link Utilization Data: We analyzed data from 138 backbone links, (these constituted most of the links in the backbone during our period of observation from September 2000 to June 2001) which were pruned from a total of 3920 uni-directional links in the complete Sprint network. For each link, we collected the SNMP link utilization values, which are obtained from the Cisco MIB variables[4] locIfInBitsSec and locIfOutBitsSec from the Cisco MIB interface group, and correspond to OIDs 1.3.6.1.4.1.9.2.2.1.1.(6,8) respectively. This link utilization is calculated as an exponential weighted moving average (EWMA) of the link load and is calculated as follows:

$$U_{new} = (U_{old})e^{-\frac{t}{360}} + U_s(1 - e^{-\frac{t}{360}}) \quad (1)$$

where, U_{new}, U_{old} are the current and previous value of the EWMA and U_s is the average link utilization sampled over every time interval of length t. Our measurements were done on routers which sampled the link utilization once every $t = 10$ seconds. As a consequence, the EWMA average on these routers can take up to

$$T = \frac{-\ln(0.1)}{10/360} = 83 \text{ samples, or 14 minutes} \quad (2)$$

to reach within 90% of their instantaneous value (for example a change in link utilization from 10% to 100% or vice versa). Since we poll the EWMA value from each router only once every 5 minutes, it may take up to three polling periods or 15 minutes to reach within 10% of the actual link utilization value. Also note that because of the smoothing effect of the EWMA, peaks in link utilization for extremely short periods (say a few minutes or less) may be completely missed in our measurement. As a consequence, our measurements on peaks in link utilization are highly conservative.

Router and Link Configuration Data: This data is collected from individual routers, and gives a detailed breakdown of the interfaces connected to each router, the locations of the two end points of the link, the capacities of the links, etc. Most of this data can be obtained using the *show interfaces* command on the router command line interface of Cisco routers. We use the configuration data to build the network topology and to correlate the SNMP link utilization data to routers and PoPs.

IS-IS Link Weights*:* We use an internal web-based network reporting tool to obtain the intra and inter-PoP link weights used in IS-IS routing on the Sprint network. These link weights (in conjunction with the router and link configuration data) are

4. A detailed description is available at http://www.cisco.com/univercd/cc/td/doc/cisintwk/ito_doc/snmp.htm

used to accurately simulate our algorithms presented in Section IV.

2) Data Analysis:

Types of Link Measured*:* We are only interested in understanding the properties of IS-IS routing over the backbone and therefore we prune links which are not relevant to our discussion. We classify the links into four major types as follows —

1) *Inter-PoP Backbone Links*: These are links which connect one backbone PoP to another backbone PoP (see Figure 10).

2) *Intra-PoP Backbone Links*: These are links within a PoP, which connect two different backbone routers in the *same* PoP.

3) *Access Aggregate Links:* These are links which aggregate customer traffic from (to) one or more metro area networks in the proximity of a single PoP and connect to (from) a backbone router.

4) *Peering Links*: These are links which connect a router in the Sprint network with a tier-1 internet service provider.

Error Pruning: An important concern regarding SNMP data is the potential for errors. Some of the key errors we saw were time periods during which data was absent, incorrect data values (e.g. link utilization is higher than capacity), the timestamp field in the SNMP data being set to zero and other sources of error. We prune all these instances of errors in the data and consider only links and periods of time which to the best of our knowledge are error free and for which both the SNMP data and configuration information are known.

Network Dynamics: We note the Sprint backbone is constantly changing and there are several links which are added, upgraded, or phased out in our period of study. Thus, not all backbone links are active at the same time and our observations over 9 months are over slowly changing topologies. We discuss these observations in the next section.

B. Utilization of the Links

We find the overprovisioning ratio (i.e. the ratio of the capacity installed in the backbone to the traffic demand) to be the same in both the intra-PoP and inter-PoP topology and that it has remained more or less constant during a period of one year. This however does not mean that the link utilization is uniform throughout the Sprint backbone. We shall show this by describing in detail the distribution of link load.

First, we study the distribution of the maximum and minimum utilizations of the backbone links at any time instant. We found as shown in Figure 1 that at any given time there is almost always a link in the backbone, which is overloaded (the X axis plots time in increments of 5 minutes). In fact there are periods where the maximum utilization of a link in the backbone goes up to as high as 90%.

On the other hand we have noticed that at all times there is always some link that has almost zero load (in the figure this overlaps the X-axis), showing the wide disparity in link utilization.

A better understanding of the distribution of link load (rather than just looking at the minimum and maximum values) can be obtained by choosing a lower utilization value[5], say 30%, and classifying links based on this utilization. In Table 1, we classify links on the basis of the fraction of time they showed a load above 30%. (The distribution would be uninteresting if we chose the load to be 50%, since most links would then never see overload.) It can be seen that at one extreme, 11% of all backbone links experience 30% load very frequently, while at the other extreme, 69% of links never experience such load even once during their lifetime.

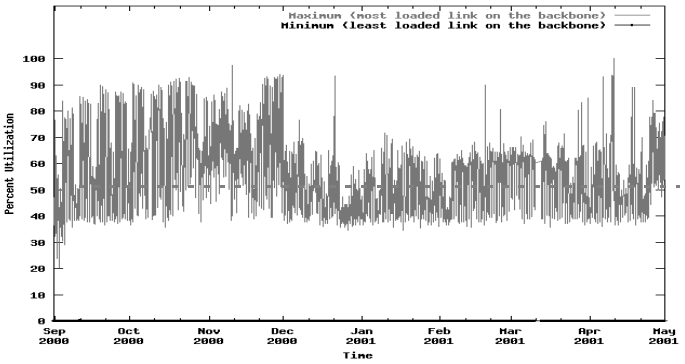


Figure 1: Maximum and minimum utilization of a backbone link at any given time.

TABLE 1 : Classification of links based on load.

Fraction of time that links showed 30% load	Percentage of Links
Never	69%
Less than 0.001	7%
Between 0.001 and 0.01	7%
Between 0.01 and 0.1	6%
More than 0.1	11%

Next, we want to know how the total load is divided amongst different links at a given time. To do this, we classify the backbone links based on their average utilization at a given instant of time. As shown in Figure 2, almost 95% of backbone links show an average utilization of below 50%, while the remaining 5% of links show overload in the Sprint network.

Since link overload is a primary concern in Traffic Engineering, we now analyze the pathology of overload.

C. Overload Characteristics of the Links.

First, we are interested in identifying those links which experience overload. We plot the number of links which exceed a load threshold during some stage of their lifetime. This is shown in Figure 3. We observe that the number of links which ever cross a specific load threshold falls off rapidly. We identify that about 11% of all backbone links get overloaded at least once during the period of observation. On average we have found that overload is rare and a typical link is overloaded for less than 0.2% of its lifetime.

[5] Though it would be more illustrative to show the exact link utilization values for all links, due to the propriety nature of some of the data, we are unable to reveal these values.

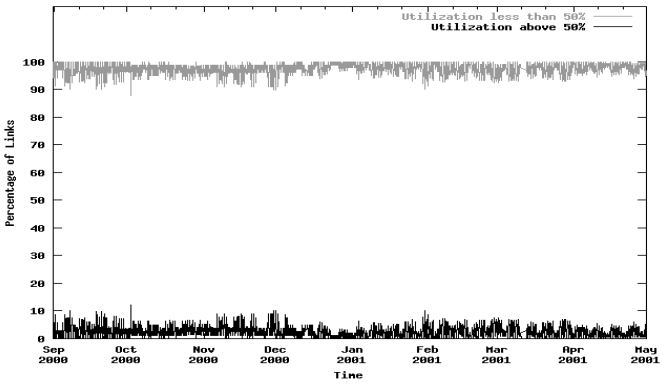


Figure 2: The percentage of links which are overloaded in the backbone at a given time instant.

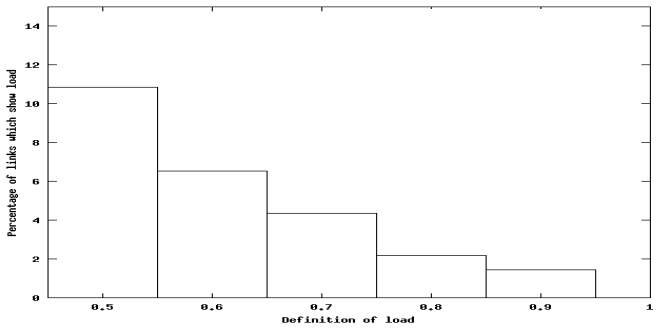


Figure 3: The percentage of links which exceed a specific load level.

Second, we are interested in knowing how many links get overloaded at a given time instant. In Figure 4, we plot the number of links which show simultaneous load. We observe that the probability of seeing more than 4-5 links which are simultaneously overloaded is negligible. This is more true when we look at periods of severe overload where link utilization reaches values of 80-90%. In fact we were unable to find any instant in time over the complete 9 month period of measurement where more than one link showed 80% load. *That tells us that at any time instant it might be possible for IS-IS to deflect traffic from such severely overloaded links to other available non-overloaded links.*

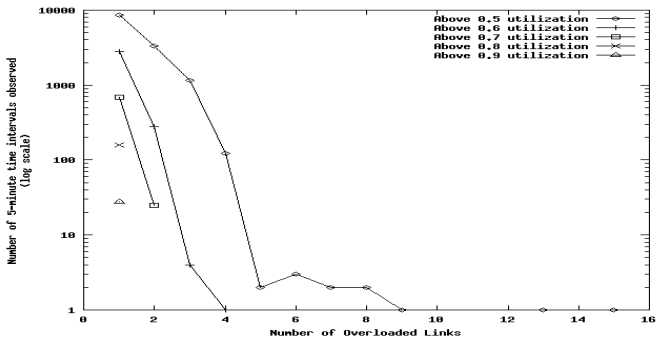


Figure 4: The number of links which show simultaneous load.

Next, we study simultaneous overload more carefully. In Figure 5, we plot all the links in the backbone which are active over the 9 month period that experience overload during their lifetime. For every time instant we mark a link "on" (i.e. it appears on the plot) if it is overloaded and "off" if it is not overloaded. We plot "on" periods for each of these links. We observe two phenomenon:

1. Horizontal alignment of points: From the figure, it is easy to see that horizontal lines correspond to links which are consistently overloaded. We found 6 such links (links numbered 6, 8, 10, 12, 13, 15 in the figure).
2. Vertical alignment of points: A vertical line which intersects many different "on" points in the figure corresponds to time instants when many links experience simultaneous overload.

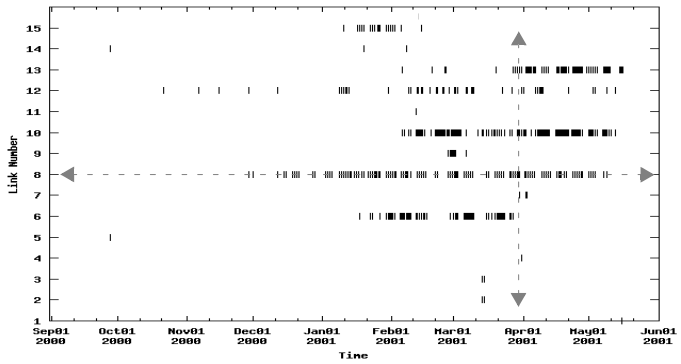


Figure 5: Occurrence of link overload for different backbone links.

We shall now give further details on the two causes of overload in the network.

1) Persistent Link Overload: The topology of the network, the traffic matrix and the nature of IS-IS shortest path routing may all potentially contribute to link overload. However, we noticed that there were links between 'popular' PoPs that experienced persistent overload even when there were other alternate paths with low utilization available. Hence, it appears that in a complex backbone, it does not seem easy to set IS-IS link weights which can balance the load across all parts in the network. In general we found that a very small percentage of links get persistently overloaded and the load on these links was usually not more than 50-65%. It is important to note that though persistent overload does not occur due to link failure, it can get aggravated during link failure.

2) Overload due to Link Failures: We classify a link as having failed if the EWMA value for the link utilization is zero.[6] We have found that more than 80% of overload occurs when it is preceded by a link failure on another link. We do this by monitoring a link which gets overloaded and checking for link failures within the previous polling period (i.e. within the last 5 minutes) for all other backbone links before we observe the overloaded link. We report two distinct types of link overload. They are as follows—

6. Unfortunately this fact by itself may not be an accurate representation of link failure.

Immediate Overload: An example of this phenomenon is shown in Figure 6, where link F shows sudden overload because traffic from alternate paths which uses links D and E are diverted by IS-IS to it. We attribute the overload on Link F due to a failure on links D and E. We are also able to verify that the load on link F is not due to transients in IS-IS route convergence. This is due to the fact that the overload on link F lasts for a period of 90 minutes whereas IS-IS converges within a few seconds [8].

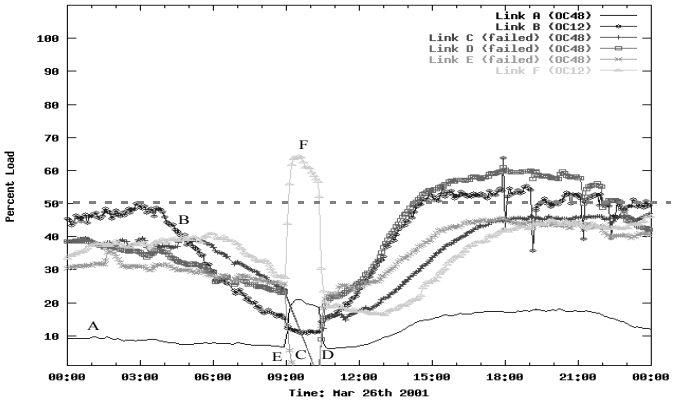


Figure 6: Immediate Overload due to Link Failure.

Slowly Increasing Overload: An example of this is shown in Figure 7. We notice that link 3 first shows immediate overload up to about 60% due to a link failure. Then its utilization climbs up from 60% to about 95% in a gradual manner over a period of six hours. We find that this is because traffic from different sources which are diverted to link 3, (due to a failure on link 2), is constantly increasing[7] due to daytime fluctuation in traffic demand. We note that if overload occurred slowly, then

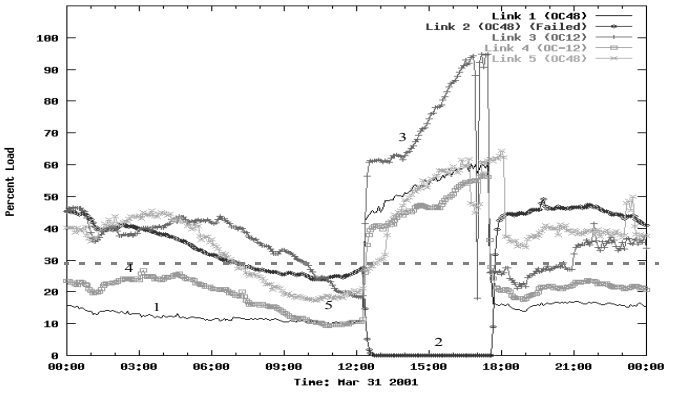


Figure 7: Immediate overload followed by slowly increasing overload.

network administrators have time to change link weights. However, this is not the case since even 'slowly increasing overload' is preceded by sudden overload. We also find that when link loads reach levels of 90% or so as shown in Figure 7, it is usually due to multiple link failures. With the advent of DWDM, multiple logical IP links can share a single fiber, and such frequent 'fiber cuts' result in multiple link failures. Also note that since in any backbone, links are constantly upgraded, there usually exist links of varying capacity, and a link failure on a high capacity link (e.g. OC48), can easily cause severe overload on low capacity links (e.g. OC12). This implies that severe overload due to link failures can be a recurring problem.

D. The Case for Deflection Routing

1) Issues with link overload: In summary, our observations show that very few links are persistently overloaded. However, the main cause of link overload is link failure. This suggests that when looking for ways to improve the performance of routing, we should concentrate on how it reacts to link failure. There are three main reasons to tackle link overload.

1. First, recall that our numbers on link overload are conservative and the actual link load over shorter intervals of time can be much higher than that measured. In case of severe overload an 80% EWMA link load could actually mean loads above 100% of the link capacity for shorter intervals of time, which may result in packets being dropped.[8]
2. Second, network operators usually upgrade links as soon as peaks in link utilization consistently reach some pre-defined threshold. Thus these peaks prevent us from increasing the average utilization of the network and result in an inefficient use of network resources.
3. Third, the advent of several interactive and real-time applications which are very sensitive to increased delay and jitter (both consequences of link overload) make it mandatory that the network be able to either prevent or immediately overcome overload.

2) The necessity of deflection routing: A number of previous proposals exist, which attempt to tackle link overload. Valiant [9] suggested that packet forwarding be done in two stages. A random intermediate node is first picked, and every packet is forwarded on its shortest path from its source node to the random intermediate node x; it is then forwarded along its shortest path between x and the destination. This has the effect of distributing the load smoothly across the network. Bak *et. al.* [10] also suggest some heuristics in choosing such a random intermediate node. However, the above method is probably overkill given the state of the backbone network that we operate in. Further, it results in very large delays, which can violate the tight end to end delay service level agreements (SLAs) on a backbone network.

There have also been a number of recent techniques, which attempt to optimize IS-IS link weights to spread the traffic over all links in a more even manner [11] or change link weights

7. We have noticed that there is wide fluctuation in the link utilization based on the time of day. Some highly utilized backbone links show almost 50-100% more load during the day than during the night and weekend hours.

8. As a rule of thumb Internet routers are expected to store about RTT = 0.25 s worth of data for TCP congestion control to work efficiently [13]. In practice, they are built to handle congestion for a much lesser period of time increasing the possibility of packet loss during overload.

during failure [12]. Similarly in [14], the authors develop a heuristic algorithm to set link weights such that the network can avoid overload during single link failures. The above techniques are well suited to tackling long term overload (for example, a fiber cut which could take hours to fix). However, our measurements reveal that most overload lasts for not more than a few minutes, and changing link weights for such a short time is unappealing. Also, since the network is complex, link overload is bound to occur inspite of careful design. Hence, there is a need for the network to react actively to overload.

In fact the problem of detecting instantaneous overload and reacting to it immediately is already in practice in telephone networks. Kelly introduced dynamic alternate routing [6] in which a network node immediately attempts to find an alternate path in case of congestion. This technique is being widely used in the British Telecom telephone network. Similarly in [7], the authors propose extending the routing protocol to calculate alternate routing paths during congestion. Indeed, our measurements point to the relevance of previous work and the necessity of deflection routing in the backbone.

Unlike [7], we are interested in a simple solution which involves no change to the routing protocols and which is distributed in nature. Next, we outline how it might be possible to design a deflection routing algorithm that can be easily and safely deployed in an operational IP backbone.

III. Requirements for the Design of a Loop-free Deflection Routing Algorithm

We will describe the conditions on the deflection algorithm, the topology of the network and the setting of link weights that satisfy the goals outlined in the previous section. We will then explore the feasibility of such a deflection routing algorithm.

A. Ensuring that the deflection algorithm is loop-free.

Definition 1: ***Strictly decreasing cost criterion:*** *Consider a packet at a node p, which forwards a packet destined to d. If the shortest path link is overloaded, then a node can deflect a packet to a neighboring node n whose cost to the destination is less than the cost of the shortest path from p to d, provided the link to n is not congested. We call this deflecting according to the 'the strictly decreasing cost criterion'. We can now state the following theorem.*

Theorem 1: Any deflection algorithm which meets the strictly decreasing cost criterion is loop-free.

Proof: This is easy to see. Assume that a packet originates at node a_1 and is destined to node a_n. Let $a_1, a_2, \ldots a_{n-1}, a_n$ be the sequence of nodes that the packet goes through. Let $W_1, W_2, \ldots W_{n-1}, W_n = 0$ be the sequence of the costs of the shortest path for each corresponding node to destination a_n. Consider an arbitrary node a_i and the next node on the path a_{i+1}. If a_{i+1} was on the shortest path from a_i to a_n then trivially, $W_i > W_{i+1}$. If not, then a_i deflected the packet to node a_{i+1}. Then the strictly decreasing cost criterion ensures that $W_i > W_{i+1}$. In both cases, the sequence $W_1, W_2, \ldots W_{n-1}, W_n$ forms a strictly decreasing sequence. Thus the deflection algorithm cannot create loops because then $W_i = W_j$ for some $i \neq j$, which contradicts the fact that the sequence $W_1, W_2, \ldots W_{n-1}, W_n$ is strictly decreasing. ❐

B. Setting link weights which meet the strictly decreasing cost criterion.

We now explore the conditions such that every node can deflect according to the strictly decreasing cost criterion. In what follows we shall see that this mandates further conditions on both the network topology and the assignment of link weights.

Definition 2: ***Neighbor Disjoint Paths*** (i, j)*: A path from node i to node j is said to be neighbor disjoint with another path from i to j, iff both paths originate from i, and have two distinct neighbors of i as their immediate next hop.*

Definition 3: ***Hop Stretch****: $H(i, j)$ The hop stretch of a pair of nodes (i, j) in a network is the ratio of the number of hops between the second shortest neighbor disjoint path and the shortest path between node i and node j.*

Note: The hop stretch is based on the length of the neighbor disjoint paths and not the weights of the paths.

Theorem 2: (Sufficiency) In a network for which every pair of nodes has a hop stretch of 1, there always exists a loop-free alternate forwarding path, if the link weights are assigned in the range $[W_{min}, W_{min}x]$, where $x \leq 1 + 1/(d-1)$, W_{min} is any arbitrary constant which denotes the minimum weight assigned to a link, and d is the diameter of the network.

Proof: Consider an arbitrary pair of non-adjacent nodes, denoted by node a_1 and node a_{g+1} in the network (We shall consider adjacent nodes in Section III.C). Let the shortest path between these nodes be of length $g \geq 2$. We shall denote the nodes in this path as $\{a_1, a_2, a_3, \ldots a_g, a_{g+1}\}$. Since the hop stretch of the network is 1, there exists an alternate path with the same length. We shall denote this by $\{a_1, b_2, b_3, \ldots b_g, a_{g+1}\}$. Note that $b_{g+1} = a_{g+1}$. We know that cost of the shortest path is given by

$$\sum_{i=1}^{g} W(a_i, a_{i+1}) \tag{3}$$

where $W(a_i, a_{i+1})$ denotes the weight of the link between node a_i and node a_{i+1}. Similarly, we have that the cost of the shortest path between neighboring node b_2 and the destination is

$$\sum_{i=2}^{g} W(b_i, b_{i+1}). \tag{4}$$

In order that the strictly decreasing cost criterion is always satisfied and using Equation (3) and Equation (4), we get that

$$\sum_{i=1}^{g} W(a_i, a_i + 1) \geq \sum_{i=2}^{g} W(b_i, b_i + 1). \quad (5)$$

The minimum value of the L.H.S is at least gW_{min}, while the maximum value of the R.H.S is at most $(g-1)W_{min}x$. Equation (5) is always true if $gW_{min} \geq (g-1)W_{min}x$ i.e.

$$x \leq 1 + 1/(g-1). \quad (6)$$

A sufficient condition which satisfies this equation is

$$x \leq 1 + 1/(d-1) \quad (7)$$

where d is the diameter of the network. ❐

As an example, consider the 6-node network shown in Figure 8. The network has a diameter of 3. If the link weights are all chosen to be within a ratio $x = 1.5$ of each other as shown, then there is always a loop-free deflection path between any two nodes.

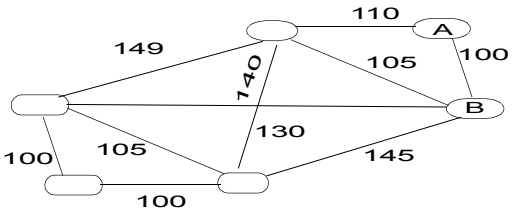


Figure 8: An example network with loop-free deflection paths.

With relation to Theorem 2, we found that in the Sprint network there are always multiple paths between every source and destination PoP. Also we found that almost all PoP pairs satisfied Equation (6). In addition, 74% of all inter-PoP link weights satisfied the more restrictive but not mandatory condition in Equation (7). We find that the case when these conditions are violated are when certain links with very low capacity (say OC3 links on an OC48 backbone) have an IS-IS link weight much larger than the average weight for other links. Nevertheless, given that Equation (6) holds for most PoP pairs, we believe that we can exploit this approach of setting link weights in the backbone.

C. Practical Considerations

There still remain now two issues which need to be resolved.

The last-hop problem: First, let us consider what happens to packets which are on their last-hop i.e. $g = 1$. In Figure 8, consider a packet at node A whose final destination is node B. The direct link from A to B is also the link with the smallest weight arising from node B. Hence there is no other node to which traffic from A destined to B can be diverted to, which has a lower cost to B than A. Clearly there are always such cases on every network for which no routing protocol (let alone deflection routing) can provide an alternate loop-free path (unless some state is maintained on the packet or router to perform source routing and prevent loops).

Note that there are actually multiple equal cost links between any two neighboring PoPs. So one would assume that with load balancing (especially IS-IS ECMP) there would never be any overload (and hence no need for deflecting packets) since the traffic between the two PoPs would be equally distributed over these multiple equal cost links from A to B preventing the scenario described above. The reason this does not happen is that multiple links between any two neighboring PoPs are usually connected via different backbone routers.

For example, consider the topologies of the backbone PoPs A and B as shown in Figure 9. These PoPs have three equal

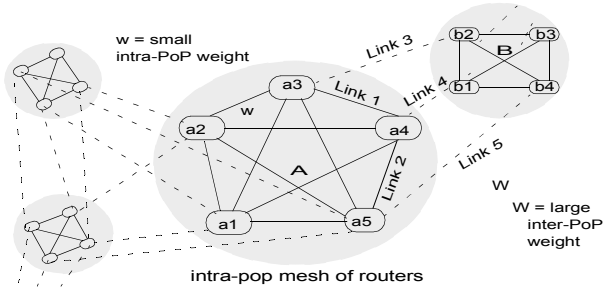


Figure 9: Topology of a Backbone Network.

cost links between them, i.e. links $a_3 - b_2$, $a_4 - b_1$ and $a_5 - b_4$. Consider the traffic which originates from router a_5 and destined to router b_4. This traffic cannot be diverted via any of the other equal cost links $a_3 - b_2$ and $a_4 - b_1$. This is because the path via these links has a higher cost, as it traverses two extra intra-PoP links. Using Sprint's real-time tool we were able to verify that this indeed occurs on the network. Amongst multiple equal cost links between two PoPs, some of them are under utilized despite having the same weight. For example, in case of overload on link $a_5 - b_4$, we will need some mechanism to move traffic to links $a_3 - b_2$ and $a_4 - b_1$. We refer to this as the 'last-hop' problem.

Issues with Intra-PoP link weights: In our discussion until now, we have not yet considered how intra-PoP links are used in routing. Let w_{max} denote the maximum weight assigned to an intra-PoP link and W_{min} denote the minimum weight on an inter-PoP link. In the Sprint backbone, the intra-PoP weights are much smaller than the inter-PoP weights, i.e. $w_{max} \ll W_{min}$. Hence the intra-PoP weights do not meet the conditions in Equation (6) and also Equation (7), and as a consequence there are pairs of routers in the backbone for which we cannot guarantee loop-free alternate paths.

This leads us to the following idea — *When there is overload, is there some way in which we can consider each individual PoP as one node (allowing the use of all the equal cost inter-PoP links connected to neighboring PoPs, and solving the last-hop problem), and perform loop-free deflection routing on the inter-PoP network (where the topology and inter-PoP link weights support the conditions for loop-free deflection)?*

This motivates the following deflection algorithm.

D. Designing a Practical Loop-Free Deflection Routing Algorithm

Consider some node p, which forwards a packet destined to node d. Let C be the cost of the shortest path and let the node on the next hop of the shortest path be m and the corresponding link be l. If link l is overloaded, then node p tries to choose some node distinct from m to deflect a packet to. This is done according to the following criteria, where $w^* = w_{max}$, the maximum weight of the intra-PoP links.

0. If the link to neighboring node (denoted by n or n') is not overloaded, then

1. Node n is chosen, if its cost to the destination is less than $C - w^*$, i.e. $C - w_{max}$.

2. Else, it can choose node n'

a. whose cost is no more than w_{max}

b. and the cost of node n' to the destination is no more than $C + w_{max}$,

c. and whose next hop to the destination d is not p itself;

The deflection routing algorithm distributes the load equally across all links to nodes n which satisfy criterion 1. In case there are no such nodes, it distributes the load across all links[9] to nodes n' which satisfy criterion 2. If there are no such nodes n, n' available for deflection, then the packet is forwarded along its shortest path.

Now consider criterion 2 in the deflection routing algorithm. Criterion 2a allows the packet to be deflected only amongst the routers within the same PoP. Criterion 2b enforces that amongst the routers which meet criterion 2a, none of them are too far away from the destination[10], and criterion 2c prevents a packet from being deflected back on the interface from which it came.

In summary, criterion 2 is in keeping with the way a single PoP is designed on the Sprint network i.e. it is a full mesh and packets get to use whatever interfaces are available on any of the routers in the PoP to get to their destination. Criterion 2 is also meant to solve the 'last-hop problem' described in the previous section. Now we are ready to state the following theorem:

Theorem 3: The deflection algorithm has no inter-PoP loops.

Proof: Suppose that a packet enters a PoP on some router and the cost of that packet to that destination is C. Since the PoP is a full mesh, all the routers in the PoP have a cost to the destination which is no more than $C + w_{max}$. Hence once a packet enters a PoP the cost of the packet to the destination can increase by no more than w_{max}.

If a packet leaves a PoP and goes to another PoP, then it can do so only by being deflected according to criterion 1 or by following the shortest path. In the former case, the cost of the packet to the destination decreases by at least $w^* + 1 = w_{max} + 1$, while in the latter case it decreases by exactly W_{min}. Since $W_{min} \gg w_{max}$, in both cases the cost of the packet to the destination decreases by at least $w_{max} + 1$ .when it goes from one PoP to another. It is easy to see that if $P_1, P_2, \ldots, P_i, \ldots, P_n$ denotes the sequence of shortest path costs to the destination for the packet when it traverses PoPs $1, 2, \ldots, i, \ldots, n$, then the sequence satisfies the strictly decreasing cost criterion. From Theorem 1, we know that there can be no inter-PoP loops. ❐

We note that it is possible for a packet to loop inside a PoP if there are multiple routers in the PoP which are overloaded. However, based on our measurements on overload, we find that this would be extremely rare.

A possible solution to completely prevent routing loops is, if a router can set the hippity bit in the IS-IS header whenever it detects overload. The hippity bit (presently set manually by network operators) is used to convey to other IS-IS routers not to send any traffic to it. In this case the deflection routing algorithm can be modified so as to never deflect to routers which are overloaded. This would completely eliminate any possibility of routing loops in the network.

Extending the algorithm to handle multiple link failures: Since link failures are the primary cause of overload, our deflection algorithm must also be loop-free when one or more failures occur. Note that when failures occur it is possible that in the worst case in a PoP with n routers, two routers in a PoP may have a cost of $(n-1)w_{max}$ between them. So if a packet is deflected multiple times within a PoP, its cost to the destination may increase by $(n-1)w_{max}$. In this case the deflection algorithm can be loop-free across PoPs by setting $w^* = (n-1)w_{max}$.[11]

Generalizing the algorithm to any network: Note that the deflection algorithm does not mandate that the PoPs require a fully meshed topology (though it will have a wider choice of routes when PoPs are fully meshed). Also note that criterion 2, can be completely eliminated in any network that has sufficient connectivity to always guarantee paths that meet criterion 1.

IV. Performance Analysis

A. Simulation Parameters

We describe the results of our simulations for the deflection routing algorithm proposed in the previous section. Simulations were performed using the Mars [15] routing simulator, on the Sprint US backbone network as shown in Figure 10. The network consists of 17 major PoPs[12]. There were a total of 70

[9] This equal distribution of load across links can be done on a flow by flow basis so as to minimize packet re-ordering.

[10] If link weights were set symmetrically on both directions of a link, then criterion 2b is made redundant by criterion 2a.

[11] This also requires setting $W_{min} > (n-1)w_{max}$, which is easily satisfied, since $W_{min} \gg w_{max}$, and n, which is the size of the PoP, is not very large. In particular this condition was always satisfied in the Sprint network.

backbone routers in our simulations, and the routers within each PoP were connected in a fully meshed topology. The backbone link speeds were set to 2.5Gb/s. The average propagation delay for inter-PoP links was set to 10ms and 1ms for intra-PoP links in accordance with measurements done in [16]. Also the processing delay on each router was set to 1ms as reported in [17]. The buffer size on each interface was to set to 32Mb (which is the default size on the Cisco IP backbone routers) and the average packet size used in our simulations was 500 bytes.

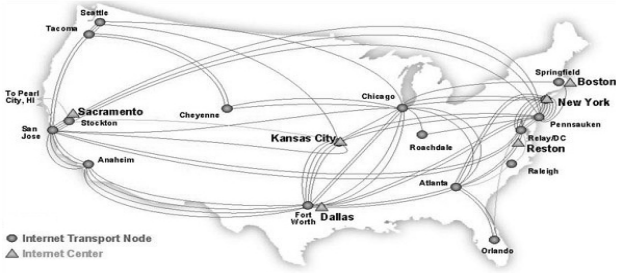


Figure 10: Topology of the Sprint IP Backbone network.

Generating a sound traffic matrix: Unfortunately due to the lack of measurement infrastructure we do not have an accurate model of the traffic matrix i.e. the amount of traffic which originates in one PoP and sinks at another PoP. One option to estimate the traffic matrix is to use techniques suggested in previous work [18][19][20]. These methods rely on SNMP link counts as the only concrete information about the network. However, we chose to use gravity models in estimating the traffic matrix as suggested in [21]. In its simplest form (as specified in Equation 10 in [21]) the amount of traffic sent from PoP i to PoP j, denoted by X_{ij}, can be specified as:

$$X_{ij} = F_i \alpha_{ij}, \tag{8}$$

where F_i is amount of traffic generated by PoP i (we obtained F_i by measurements on the access aggregate links for each PoP), and α_{ij} is a fraction based on the relative capacities C_i (sum of the capacities of all incoming access aggregate links to a PoP) of the PoPs i and j. Note that in our simulations we would like to study how the network would perform under varying link utilization. For us to proportionately scale the traffic matrix, we define a parameter called 'normalized load' λ on the network. The normalized load for the Sprint network λ_s is defined as [13]:

$$\lambda_s = \frac{1}{N} \sum_{i=1}^{N} \frac{F_i}{C_i}, \tag{9}$$

where N is the number of PoPs. In order to generate a traffic matrix with normalized load λ, we re-write Equation (8) as

$$X_{ij} = \left(\frac{\lambda}{\lambda_s}\right) F_i \alpha_{ij} \tag{10}$$

In the following simulations we consider the performance of the network when overload occurs both in the presence and absence of deflection routing. Our simulations are for 20 million timeslots. In all these experiments we measure network statistics every 1000 timeslots, giving us 19,900 periods of observation (we ignore the first 100,000 time slots to eliminate any transients in the network). As we have seen, link failures contribute to more than 80% of all overload in the network. So we will focus on studying the performance of the network during link overload caused by both single and multiple link failures. We generate link failures (both single link and fiber cuts) with a uniform distribution with a mean failure time of 100,000 timeslots. Note that since we observe the network during a single failure or a single fiber cut, and measure the relative performance of the network with and without deflection routing, the actual duration of the failures or fiber cuts are unimportant in these simulations. In all our simulations, the routers maintain an EWMA of the utilization on each link, similar to that in Equation (1). We shall refer to this as the SNMP-EWMA.

Reacting to Overload: Note that in our simulations if a link reacts to overload by measuring the SNMP-EWMA, then it would be too late to prevent short term overload. This is because the SNMP-EWMA value can take up to 15 minutes to reach within 10% of the link utilization value, and most overloads last for a few minutes. Thus the router would completely miss seeing this overload! Hence, a correct choice of a moving average of link utilization is critical to identifying and alleviating transient overload. As a consequence in our simulation we maintain another EWMA (called FAST-EWMA) which gives higher weight to the instantaneous link utilization value than that in Equation (1) and converges to 90% of the observed link load within 10 samples (10000 timeslots) of the simulation.[14] A link in the simulations would begin deflecting packets if the FAST-EWMA crosses a threshold value. Since we want the routers to prevent overload (i.e. 50% link utilization), the threshold value for deflection is set slightly below that to 45%.

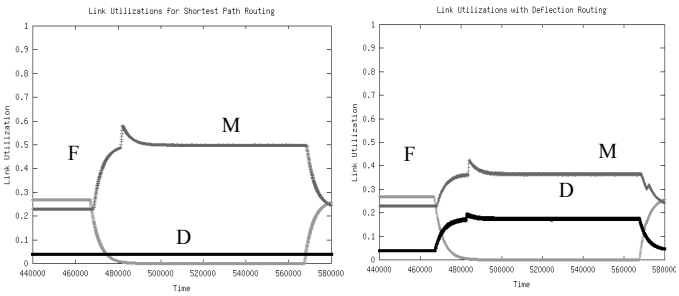
B. Overload due to a single link failure

In related work, we have ascertained that single link failures contribute to more than 45% of all failures [8] and hence we begin by studying single link failures. We conduct a number of experiments where a link is sent into overload because shortest path routing diverts traffic from a path with a failed link to a new path, sending some link on the new path into overload. We will illustrate this with a typical example of a single link failure. Figure 11a illustrates a link failure with shortest path routing. The Y-axis plots the SNMP-EWMA of the link utilization. Link F is an example of a failed link (which carried a load of 28% before failure). When link F fails, shortest path routing

[12] For proprietary reasons the exact topology is not shown.
[13] Again, due to due to proprietary reasons we cannot reveal λ_s.

[14] Comparing this to Equation (1), we can see that on a Cisco router which samples link utilization every 10 seconds, FAST-EWMA converges to 90% of the instantaneous link utilization within 100 seconds.

diverts traffic via an alternate path, causing some link M on the alternate path to become overloaded. Note that there is another link D (attached to the same router that has link M), which is lowly utilized. With deflection routing as shown in Figure 11b, part of that traffic is deflected to link D. Link D experiences a load of around 18% during failure. One can see that the link M, which reached a utilization of around 51% during failure with shortest path routing, does not experience more than 40% load with deflection routing. Note that because the links react to the



(a) (b)

Figure 11: A controlled experiment with a single link failure.

FAST-EWMA, the routers react to overload quickly.

We are interested in understanding how the network performs as the normalized load increases. In order to study overload which occurs only due to link failures, we first simulate the network in the absence of link failures. We find that when the normalized load λ is less than 0.3, there is no overload in the absence of link failures. Hence in these experiments we only consider the case when $\lambda \in \{0, 0.3\}$. The results from our simulations are shown in Figure 12. On the X-axis, we plot the normalized load λ of the network. On the Y-axis, we plot the average number of instants of overload (where each overloaded link individually contributes to an overload instant at a given time) observed in the network. With shortest path routing, link failures lead to overload as soon as $\lambda > 0.2$ on an average. However, with deflection routing there is almost no overload. We found that for 94% of cases of single link failures,

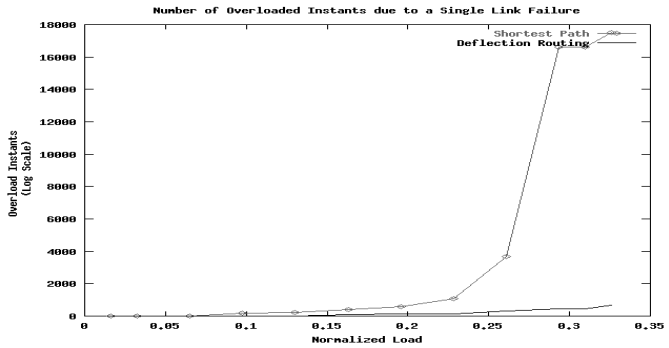


Figure 12: Link overload during a single link failure.

the deflection routing algorithm was able to compute a loop-free alternate path. This is not surprising given the rich number of alternate paths of equal length (which are candidates for loop-free deflection) in the Sprint backbone as shown in Figure 10. When such a path was not available, the router was able to find another router within the same PoP which met criterion 2 in the algorithm.

We would like to know how the maximum load on the network varies as a result of deflection routing during link failure. In Figure 13, we plot the maximum load experienced for different values of λ. Note that though the deflection threshold is set to 45%, the SNMP-EWMA link utilization is capped to as low as 38-40% because the routers react to load as reported by the FAST-EWMA.

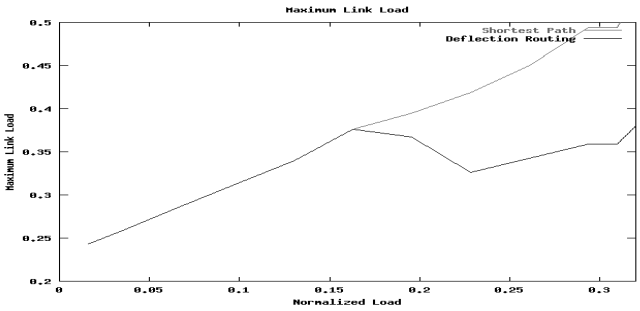
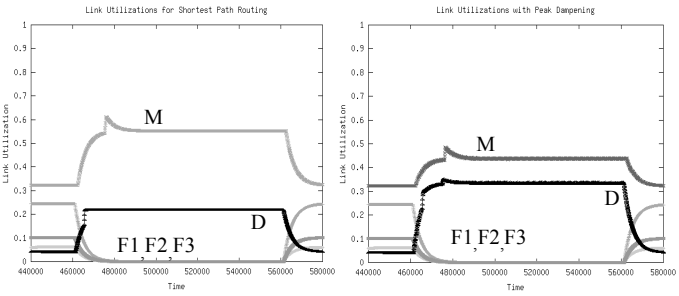


Figure 13: Maximum load on a link in the network during link failure

C. *Overload due to fiber cuts*

A second important cause of link failure is due to a fiber cut. We note that this can lead to the simultaneous failure of two or three links (which share the same fiber) from a single PoP. Recall that fiber cuts usually cause severe overload (where the link utilization can reach as high as 80%) and hence this is of particular interest to us.

We shall again begin by explaining a typical example of a fiber cut scenario. Consider Figure 14a where are three links



(a) (b)

Figure 14: A controlled experiment with a single fiber cut

F1, F2, F3, which originate from the same PoP and share the same fiber. During a fiber cut, these three links fail simultaneously. With shortest path routing, traffic from links F1, F2 and F3 is diverted to links D and M. In this process, the utilization of link D increases from 5% to 21%, whereas link M gets overloaded and shows 55% utilization. Now compare this to Figure 14b with deflection routing. In this case, part of the traffic from link M is deflected to link D. The utilization of link D rises to 30%, whereas link M utilization is capped at 43%. Note that link D's utilization increased from both traffic diverted due to the recomputation of the shortest path during failure as well as traffic which is deflected to it.

We also plot the occurrence of link overload in the presence of fiber cuts for varying normalized load in Figure 15.

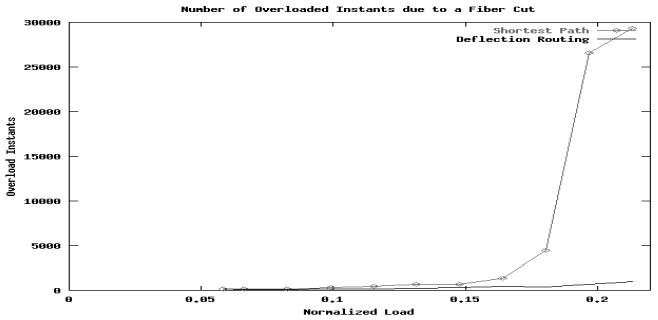


Figure 15: Link overload during a single fiber cut

It is interesting to compare this to Figure 11. As expected the network begins to experience overload at lower values of λ as compared to the case of a single link failure. Note that in the Sprint network, fiber cuts still leave the network connected to one (sometimes two) PoP(s) with multiple equal cost links to these PoPs. In 53% of all fiber cuts, the network was able to calculate a loop-free alternate path, whereas the remaining 47% of the time packets were diverted to routers within the same PoP. However, the fully meshed nature of the PoPs allows the network to be quite resilient to single instances of a fiber cut. We plot the 'peak dampening' effect of deflection routing during fiber cuts, for varying normalized load in Figure 16.

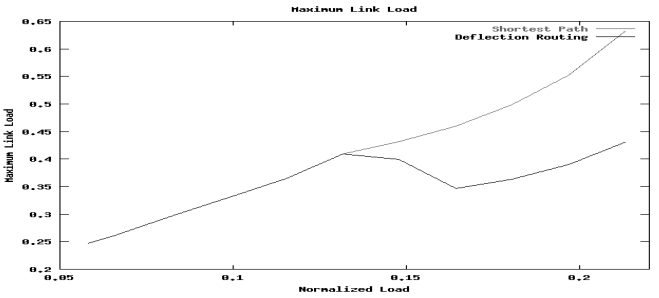


Figure 16: Maximum load on a link during a single fiber cut

D. Costs associated with deflection routing.

The main cost associated with deflection routing is the increased propagation delay. In our simulations we found that the maximum additional delay faced by packets as a consequence of deflection routing was 12ms.

V. Conclusions

We have studied the pathology of link overload in the network and have proposed a deflection routing algorithm to alleviate overload by exploiting the highly meshed nature of the the backbone and a judicious use of link weights. Our proposal is based on the Sprint network which is built using a pure IP philosophy, though it can be applied to other networks, say those enabled with MPLS. The algorithm is distributed and is loop-free across the PoPs. However, we note that this technique cannot be used if the backbone is not sufficiently meshed to allow at least two equal length paths between PoPs. Also a loop-free deflection algorithm does not ensure stability, since deflection itself can cause instability if there are too many instances of overload — something which can be quite common if the backbone shows high average utilization. However, based on our current observations on backbone topology and link utilization, we believe that our current assumptions are practical, and that our technique can be effective in tackling link overload.

VI. Acknowledgements

The authors would like to thank Nick McKeown and Frank Kelly for insightful discussions.

VII. References

[1] D. P. Bertsekas, *Linear Network Optimization: Algorithms and Codes*, The MIT Press, 1991.

[2] S. Vutukury and J. J. Garcia-Luna-Aceves, "MDVA: A Distance-Vector Multipath Routing Protocol", In Proceedings of the IEEE INFOCOM, pages 557--564, 2001.

[3] S. Vutukury and J.J. Garcia-Luna-Aceves, "An algorithm for multipath computation using distance-vectors with predecessor information", Proc. of ICCCN, Oct. 1999.

[4] W. T. Zaumen and J.J. Garcia-Luna-Aceves, "Loop-Free Multipath Routing Using Generalized Diffusing Computations", Proc. IEEE INFOCOM, March 1998.

[5] Multipath Issues in Unicast and Multicast Next-Hop Selection, available at http://www.ietf.org/rfc/rfc2991.txt

[6] F. P. Kelly, "Network routing", *Proc. R. Soc. Lond. A*, 337:343--367, 1991.

[7] Z. Wang and J. Crowcroft, "*Shortest Path First with Emergency Exits*," Proc. ACM SIGCOMM, Philadelphia, PA, Sep. 1990, pp. 166-176.

[8] G. Iannaccone, C. Chuah, R. Mortier, S. Bhattacharyya, C. Diot, "Analysis of link failures in an IP backbone", to appear in Proc. IMW Marsielle, France, 2002.

[9] L. G. Valiant, *A Scheme for Fast Parallel Communication*, SIAM Journal on Computing, Vol. 11, No. 2, May 1982.

[10] S. Bak, J. A. Cobb, E. L. Leiss, *Load-Balanced Routing via Bounded Randomization*, Proc. 11th IASTED International Conf. on Parallel and Distributed Comp. and Systems, pp. 857-862, Boston, MA, Oct., 1999.

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

[12] B. Fortz and M. Thorup, "Optimizing OSPF/IS-IS Weights in a Changing World," *To appear in IEEE JSAC Special Issue on Advances in Fundamentals of Network Management*, Spring 2002.

[13] Curtis Villamizar and Cheng Song, "High performance TCP in ANSNET," ACM CCR, vol. 24, no. 5, Oct. 1994

[14] A. Nucci, B. Schroeder, S. Bhattacharyya, N. Taft, C. Diot, "IS-IS link weight assignments for transient link failures", Sprint TR02-ATL020133.

[15] C. Alaettinoglu, K. Dussa -Zieget, I. Matta, O. Gudmundsson, and A.U. Shankar, *MaRS – Maryland Routing Simulator* Version 1.0. Department of Computer Science, University of Maryland, 1991.

[16] C. Boutremans, G. Iannaccone, C. Diot, "Impact of Link Failures on VoIP Performance", Proc. Nossdav, Miami, USA, May 2002.

[17] K. Papagiannaki, S. Moon, C. Fraleigh, P. Thiran, F. Tobagi, C. Diot, "*Analysis of Measured Single-Hop Delay from an Operational Backbone Network*", Proc. IEEE Infocom '02, New York, USA.

[18] J. Cao, D. Davis, S. Vander Weil, and B. Yu, "Time-Varying Network Tomography," J. of the American Statistical Association., 2000.

[19] O. Goldschmidt, "ISP Backbone Traffic Inference Methods to Support Traffic Engineering", In Internet Statistics and Metrics Analysis (ISMA) Workshop, San Diego, CA, December 2000.

[20] C. Tebaldi and M. West, "Bayesian Inference of Network Traffic Using Link Count Data," J. of the American Statistical Association., pp. 557–573, June 1998.

[21] A. Medina, N. Taft, K. Salamatian, S. Bhattacharyya, C. Diot, "Traffic Matrix Estimation: Existing Techniques Compared and New Directions", in Proc. ACM Sigcomm 2002, Pittsburgh, PA, USA.