A Traffic Engineering Approach based on Minimum-Delay Routing

Srinivas Vutukury
Computer Science Department

J.J. Garcia-Luna-Aceves
Computer Engineering Department

Baskin School of Engineering
University of California, Santa Cruz, CA 95064
{vutukury,jj}@cse.ucsc.edu

Abstract— **Single-path routing provided by today's widely used IGP's such as RIP make extremely inefficient usage of network bandwidth, and is evident in the large end-to-end delays flows experience in single-path routing as compared to minimum-delay routing. Enhancement to OSPF such as optimized multipath have not proved to be adequate to bridge this large delay gap. Practical implementation of minimum-delay routing, on the other hand, have been largely unsuccessful for reasons such as scalability, slow convergence and out-of-order packet delivery. This paper proposes a traffic engineering solution that for a given long-term traffic matrix adapts the minimum-delay routing to the backbone networks which is practical and is suitable to implement in a Differential Services framework. A simple scalable packet forwarding technique is introduced that distinguishes between datagram and traffic that requires in-order delivery and forwards them accordingly and efficiently. Using simulations we show that the delays obtained are comparable to minimum delays and far better than single-path routing.**

I. Introduction

The shortest-path routing protocols such as RIP[10], EIGRP[1] and OSPF[14] that are widely used as Interior Gateway Protocols (IGPs) in today's Internet are highly inefficient with respect to bandwidth usage. To improve bandwidth utilization and reduce delays, several improvement to the basic routing protocols have been proposed[15], [19], [21], [6]. However, there has been no theoretical basis for such improvement and have remained adhoc. For example, in ECMP[15] load is distributed equally over multiple equal-cost paths typically using simple round-robin distribution or using hashing on the packet headers. To make optimal use of network resources and minimize delays, traffic between source-destination pairs may often have to be split and routed along multiple paths in proportions that are not necessarily equal[9]. Though OSPF-OMP[21] suggests using unequal traffic distritbution on multiple paths, it provides no concrete method to find a distribution that conforms with theoretically optimal distribution. One practical implementation of optimal routing, namely Codex[11], uses virtual circuits to setup up flows between source-destination to realize optimal distribution, but the architecture introduces unacceptable complexity at the ingress and core routers and is not scaleble. In [22], we showed how minimum-delay routing can be approximated and adapted to highly dynamic traffic on short timescales. The main drawback of that approach was that packets may arrive out-of-order at destination effecting protocols such as TCP. This paper is an extension of that work and addresses the out-of-order packet delivery problem.

This work was supported in part by the Defense Advanced Research Projects Agency (DARPA) under grants F30602-97-1-0291 and F19628-96-C-0038.

Recently, Traffic Engineering (TE), has received tremendous attention in the Internet community and several IETF drafts have appeared [2], [18], [12]. Typically in a TE approach, for a given optimization criteria and input traffic matrix specified over medium-to-large time scales, the goal is to determine a set of flows with associated paths and bandwidths that meet the optimization criteria. TE solutions are most effective in a network under a single administrative domain such as ISPs, where knowledge of the link characteristics and input traffic matrix can be obtained. TE is usually employed in operational networks after topology and capacity planning have taken place. Traffic engineering has a wide scope and covers diverse forms of routing that address survivability, QoS, policy-based routing, apart from congestion management and bandwidth utilization. Bandwidth utilization and reduction of delays due to congestion, however, are of pressing importance and generally more difficult to address, and thus is the focus of this paper.

Despite the initial failure of direct introduction of minimum-delay routing in networks, a traffic engineering approach to minimum-delay routing seems to have great potential for success. We explore this potential in this paper. To the best of our knowledge, this is the first attempt to construct a TE system based on minimum-delay routing formulated in [9].

There are several algorithms to solve the minimum-delay routing problem[3], [17]. As in most optimization problems, a solution to the minimum-delay routing requires splitting of traffic between a source-destination pair along multiple paths. Setting up explicit paths from the source to the destination using MPLS leads to complexity at the source. On the other hand, if splitting of flows is not allowed, in order to simplify MPLS implementation, the solutions obtained are sub-optimal[23].

To obtain an accurate implementation of the solution that is practical and scalable we use adopt ideas from Differential Services model[7] and OSPF-OMP[21]. The solution to the minimum-delay problem is obtained using off-line algorithms in the form of routing parameters which are then downloaded into the routers, manually or using automated tools. The packet forwarding module to forward packets according to the routing parameters. At the ingress router, a key is inserted in each packet. In the intermediate routers packets are forwarded based on the key and the routing parameters. This method is better than hashing on source-destination pair[21] as it distinguishes between many connections between a source-destination pair. Also dif-

Report Documentation Page

Form Approved
OMB No. 0704-0188

1. REPORT DATE	2. REPORT TYPE	3. DATES COVERED
2000		**00-00-2000 to 00-00-2000**

4. TITLE AND SUBTITLE	5a. CONTRACT NUMBER
A Traffic Engineering Approach based on Minimum-Delay Routing	5b. GRANT NUMBER
	5c. PROGRAM ELEMENT NUMBER
6. AUTHOR(S)	5d. PROJECT NUMBER
	5e. TASK NUMBER
	5f. WORK UNIT NUMBER
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) **University of California at Santa Cruz,Department of Computer Engineering,Santa Cruz,CA,95064**	8. PERFORMING ORGANIZATION REPORT NUMBER
9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)	10. SPONSOR/MONITOR'S ACRONYM(S)
	11. SPONSOR/MONITOR'S REPORT NUMBER(S)

12. DISTRIBUTION/AVAILABILITY STATEMENT
Approved for public release; distribution unlimited

13. SUPPLEMENTARY NOTES

14. ABSTRACT

15. SUBJECT TERMS

16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT	18. NUMBER OF PAGES	19a. NAME OF RESPONSIBLE PERSON
a. REPORT	b. ABSTRACT	c. THIS PAGE			
unclassified	**unclassified**	**unclassified**		**6**	

ferential treatment is given to datagram and traffic that requires in-order delivery. The effect is that a more accurate distribution of actual traffic according to routing parameters is achieved and consequently the end-to-end delays are closer to the optimal. The architecture is practical and can be implemented in conjuction with other per-hop behaviors the Differential Services architecture.

The paper is organized as follows. Section II describes the minimum-delay routing problem and the challenges involved in implementing it. Section III describes our implementation strategy. In Section IV, through simulations we evaluate the effectiveness of the TE system in reducing end-to-end delays and congestion. Section V concludes the paper.

II. Minimum-delay routing

We use the *minimum delay routing* or *optimal routing* formulated in Bertsekas and Gallager [4] as the basis for our traffic engineering technique. The formulation is reproduced here for convenience. In the next section we will describe our implementation.

Let a computer network be represented as a graph $G = (N, L)$ where N is the set of routers and L is the set of links between them. N^i is the neighbor set of router i. For $i \neq j$, let $r_j^i \geq 0$ be the expected input traffic, measured in bits per second, entering the network at router i and destined for router j. Let P_j^i be the set of all directed paths connecting i and j in the network, and let $W = \bigcup P_j^i$. For $p \in W$, let $x_p \geq 0$ be the traffic rate on path p. IF f_{ik} is the expected traffic, measured in bits per second, on link (i,k), where $0 \leq f_{ik} \leq C_{ik}$ and C_{ik} is the capacity of link (i,k) in bits per second, from conservation of traffic we have

$$r_j^i = \sum_{p \in P_j^i} x_p \quad (1)$$

$$f_{ik} = \sum_{(i,k) \in p, p \in W} x_p. \quad (2)$$

Let $D_{ik}(.)$ be defined as the expected number of messages or packets per second transmitted on link (i,k) times the expected delay per message or packet, including the queuing delays at the link. We assume $D_{ik}(.)$ depends only on flow f_{ik} through link (i,k) and link characteristics such as propagation delay and link capacity. We assume $D_{ik}(f_{ik})$ is a continuous and convex function that tends to infinity as f_{ik} approaches C_{ik}. This is the case for example when the link is modeled as an M/M/1 queue. The total expected delay per message times the total expected number of message arrivals per second is denoted by D_T and defined as

$$D_T = \sum_{(i,k) \in L} D_{ik}(f_{ik}) \quad (3)$$

The minimum-delay routing problem is to minimize D_T. This is a non-linear programming problem, and the solution is the set of flows : $\{x_p > 0\}$. There are many algorithms to solve this problem [3], [5], [8], [9], [13], [20], [16], [17], and any of them can be used to obtain the solution. If necessary, a fast but suboptimal method described in [22] can also be used. Once the solution is obtained, mechanisms such MPLS's LSP, virtual-circuits or routing parameters are setup in the network to forward traffic along paths specified by the solution. Our proposed implementation is based on routing parameters and is described in the next section, but before we move on to the next section let us see how the above solution can be implemented using MPLS and virtual-circuits and what the drawbacks are.

We assume the TE solution is implemented in a single contiguous domain where the traffic matrix and the link characteristics is known apriori. The minimum-delay routing algorithm is first applied offline on the given traffic matrix (the set $\{r_j^i\}$) and the given network using any of the algorithms in the references mentioned above to obtain the flow set $\{x_p > 0 | p \in W\}$. For each flow x_p, an MPLS label and the associated bandwidth is obtained and a corresponding LSP (label-switched path) is setup in the network. The end result is, at each source node a set of labels with corresponding bandwidths are obtained for each destination. Each source node then distributes the traffic originating at the node for each destination according to the bandwidth and assigns labels to the packets. The main drawback of the above technique is that at each source for each destination there can be $0(L)$ flows. Therefore there can be potentially L flows as input to the weighted round-robin (WRR) procedure that distributes traffic for a the same destination. This quadratic complexity can restrict the scalability of MPLS method. In the interior routers too, the number of lables increases rapidly, but this can be controlled to certain extent if multipoint-to-point aggregation of LSPs is used. A virtual circuit implementation of the solution, such as Codex[11], has the same scalability problem.

III. Proposed Implementation

The complexity in the MPLS and virtual-circuit implementation can be overcome if the flows are setup using routing parameters. The routing parameter ϕ_{jk}^i specifies the fraction of the traffic that i receives for destination j that it has to forward on link (i,k), and is defined as follows[4]:

$$\phi_{jk}^i = \frac{f_{ik}(j)}{\sum_{m \in N^i} f_{im}(j)} \quad (4)$$

where $f_{ik}(j)$ is the rate of traffic destined for j that flows on link (i,k).

Neighbors for which the routing parameters are greater than zero are successors, and the set is defined as $S_j^i = \{k | \phi_{jk}^i > 0\}$. Also note that $\phi_{jk}^i \geq 0$ and $\sum_{k \in N^i} \phi_{jk}^i = 1$. The next-hop set S_j^i represent a directed-acyclic graph. Observe that if ϕ's are restricted to 0 or 1, the routing problem reduces to single-path routing.

Once the routing parameters are obtained, they are downloaded into the routers. The packet forwarding mechanism of the routing architecture then ensures that traffic is forwarded according those parameters. The WRR handles atmost N^i inputs for each destination in this case, which is far more scalable than $O(L)$. This task is non-trivial for two reasons: non-zero

packet-size and the requirement of in-order delivery of packets. The WRR distributor is inadequate to handle this situation and a more sophisticated mechanism is needed. In the rest of the section we describe this mechanism. We assume that there are two types of traffic: one tolerates out-of-order packet delivery (e.g.,UDP) and the other does not (e.g.,TCP). (Note that we will denote packets that do not require in-order delivery using UDP and those that require in-order delivery with TCP) Because TCP traffic must be delivered in-order, granularity of allocation for TCP traffic is at the flow level, but for UDP traffic it is at packet level. In the presence of only UDP traffic, achieving accurate traffic distribution according routing parameters can be easily done using WRR. To delivery TCP traffic in-order, a straightforward method that is often suggested is using a hashing function on the packet's source-destination address to determine the next-hop[21], [19]. In OSPF-OMP[21], only the source-destination pair is used in the CRC16 hashing function which gives only a coarse distribution. By using TCP port numbers in addition to source-destination addresses in the hash function, finer distribution can be achieved. But this can be expensive to implement because unpacking of the packets is required at each hop. Also in OSPF-OMP, both UDP and TCP traffic are handled identically. We show that by treating UDP and TCP traffic differently and accounting for different packet sizes, greater fidelity between routing parameters and actual traffic distribution can achieved. In our approach, the hashing is decoupled from the source-destination addresses, and hybrid packet forwarding is used to improve granularity in distribution.

A. Forwarding Datagram traffic

Before proceeding to describe TCP traffic and hybrid packet-forwarding, we will show that, if only UDP traffic is present and L is the maximum packet size, traffic can be forwarded such that atmost L amount of traffic more than the traffic that would be allowed by the routing parameters. This limitation imposed by the packet-size is fundamental because packet transmission is non-preemptive.Fig. 1 below describes the procedure for forwarding when only UDP traffic is present.

---

procedure datagram-forwarding(P)
{*Executed at* i *on arrival of the packet P for* j.}
begin
For some $k \in S_j^i$, let $W_{jk}^i \leq \phi_{jk}^i W_j^i$;
Forward P to neighbor k;
$W_{jk}^i \leftarrow W_{jk}^i + sizeof(P)$;
$W_j^i \leftarrow W_j^i + sizeof(P)$;
end

Fig. 1. Packet forwarding when out-of-order packet delivery is acceptable.

It can be viewed as a generalization of single-path routing; in single-path routing all packets are forwarded to a single next-hop, whereas here, packets are forwarded to each of the several next-hops in proportions specified by the routing parameters. This procedure is extended later to handle hybrid traffic. In Fig.1, W_j^i is the total traffic that node i receives and forwards for j, and W_{jk}^i is the portion of that traffic that is forwarded to neighbor k. When a packet for destination j is received, the node first checks which neighbor k has a deficit and forwards it to that neighbor, after which W_j^i and W_{jk}^i are updated accordingly. The procedure allows at most L bits in excess than what the routing parameters allow. This is the finest distribution one can obtain without breaking packets into smaller units. The proof is provided in the appendix.

B. Forwarding TCP traffic

For TCP traffic, because the granularity is at flow level, such fine distribution of traffic as in UDP is very difficult, but by making some reasonable assumptions, a fairly accurate distribution can be achieved. We assume there are sufficiently large number of flows passing through a router and the number of flows is several order of magnitude greater than the number of next-hop choices. The duration of TCP connections, the average rates of TCP connections and the packet sizes are all uniformly distributed so that bandwidth of a group of TCP connections is proportional to number of connections in the group. In backbone networks where there are large number of flows, we believe this assumption is quite reasonable. When the number of flows is low, however, the distribution is relatively imprecise. But this should be acceptable because when network load is low delays due to congestion not a serious problem. We will proceed with these assumptions, and present a procedure for TCP packet forwarding. In section IV, we will give some experimental results to validate some of these assumptions. It should be noted that under heavy load condition the performance of the routing scheme in the worst case only drops to that of single-path routing.

As mentioned earlier, we use an architecture similar to the Diffserv model and obvious advantage is it can be incorporated along with other differentiated services. At the ingress router, for each TCP connection a randomly generated key is associated. The ingress router maintains this per-connection table. When a packet of that connection arrives, the key is inserted into the packet. Effectively the computation of hash is decoupled from the source and destination addresses. The codepoint (TOS or DSFIELD field) of the packet indicates, among other things, that the packet is of TCP-type. Within the core the key is used to hash into the next-hop. The 13-bit fragment offset is used to hold the hash key. To prevent fragmentation in the domain, the DF field is set. This is possible because this solution is applied in a single domain and it is feasible to know the MTUs of all the links. At the ingress node, use of the 13-bit offset field can thus be prevented. The per-hop behavior related to queue management can still be implemented in conjunction with minimum-delay routing. By decoupling the key from the source-destination, better randomness in the key distribution can be achieved and the per-packet processing at each hop can be reduced. We must mention that per-connection table can be eliminated and CRC16 hash be used on each entering packet instead. But remember that Diffserv architecture maintains some per-connection information for profiling and other purposes.

In OSPF-OMP, boundary values are associated with each next-hop. We propose a different method which uses more memory, but is faster. With each destination j at node i, a hash table, denoted by HT_j^i, is associated. The table is a single column table with M_j^i entries. With each entry of the table

a next-hop $k \in S_j^i$ is associated. The next-hops are distributed randomly over the range, and the fraction of entries that point to a particular next-hop will be proportional to the routing parameters. That is, for each $k \in S_j^i$, $m_{j,k}^i$ entries of HT_j^i, chosen randomly, are filled with the value k, where $m_{j,k}^i = \phi_{j,k}^i M_j^i$ and $M_j^i = \sum_{k \in S_j^i} m_{j,k}^i$. In other words, each entry in the hash table points to some neighbor in the successor set and the number of entries filled with a successor is proportional to the routing parameters. Comparisons with boundary values in OSPF-OMP has $O(N^i)$ complexity.

When a TCP packet for j arrives at i, the mod function on the key is used to index into the table HT_j^i to obtain the next-hop. If M_j^i is chosen such that it is a power of two, the lower $log_2(M_j^i)$ bits of the key can be used to index into the hash table. Because of the assumptions made, this should result in each next-hop receiving the amount of traffic in accordance with the routing parameters.

C. Hybrid packet forwarding

Because of lack of complete control on the packet forwarding of TCP traffic, the actual traffic forwarded can deviate significantly from the amounts specified by the routing parameters. The skew in the distribution introduced by the hashing mechanism, can be ironed out to some extent if there is some UDP traffic also present. The UDP packets then can be forwarded to neighbors that received too little traffic compared to what the routing parameters allow. OSPF-OMP does not make this distinction. The modified packet forwarding procedure is as shown in the Fig.(2).

---

procedure hybrid-forwarding(P)
{*Executed at i on arrival of packet P for j.*}
begin
  if (P is a UDP packet) **then**
    For some $k \in S_j^i$, let $W_{jk}^i \leq \phi_{jk}^i W_j^i$;
  endif
  if (P is a TCP packet) **then**
    Let P's header map to next-hop k;
  endif
  Forward P to neighbor k;
  $W_{jk}^i \leftarrow W_{jk}^i + sizeof(P)$;
  $W_j^i \leftarrow W_j^i + sizeof(P)$;
end

Fig. 2. Packet forwarding in the presence of both UDP and TCP packets.

The hybrid procedure is similar to the UDP-only forwarding procedure described in Fig.1, except that the skew in distribution created by TCP traffic is mitigated by UDP traffic; greater the share of UDP traffic in the total traffic, finer is the distribution of the traffic according to the routing parameters. In the diffserv model, out-of-order profile packets can be treated as datagram traffic. In section IV, we will give some performance figures regarding this. The amount of extra space required in the routing table is of the order of $O(NMN^i)$. The processing time required is of the order $O(log(N^i)$. No CRC16 hash at each hop is used and no comparison with boundary values is done. We believe that the proposed TE architecture can be easily deployed in existing networks. Though hashing is not new, when combined with Diffserv framework and hybrid packet forwarding can give significant benefits.

The architecture described above can be implemented with current IP forwarding technologies with small modifications and in the Diffserv framework without needing other forwarding technologies such as MPLS. It can be used to implement other traffic engineering approach using other optimization criteria [23]. The only requirement is that the solution should be representable using routing parameters.

IV. Experimental Results

We tested the effectiveness of our TE scheme through a series of simulation experiments. In each of the experiments, for the same given input traffic matrix and network configuration, the routing parameters are first computed using an offline minimum-delay routing algorithm and downloaded into the router tables. After that packets are forwarded according to the routing parameters. The end-to-end average delays are measured for each flow. Comparisons are made between delays obtained for single-path routing, different volumes of TCP flows, and different proportions of TCP and UDP traffic.

The network used in the simulation is shown in Fig.3. The network is a contrived network with node degree high enough to provide several next-hop choices, but low enough so that there are not too many one-hop paths. The links have bandwidth 5MB, propagation delay of 100 microseconds and packets are of size 1000 bytes.

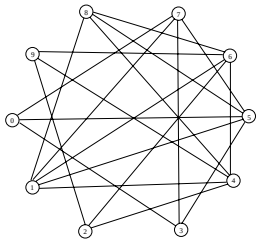

Fig. 3. Topology used in simulations

Experiment 1: The delays of SP are compared with the delays obtained in our TE scheme. The input traffic consists of 500 identical TCP flows; 50 of them originating from each node. Fig.4(a) makes the delay versus load comparison. The x-axis denotes the load and the y-axis the delay. The average delays of single-path routing are denoted by SP and the delays of our scheme are denoted by TE. Observe that for a given average delay, the load that the network can carry is much greater in TE scheme. At very low loads SP performed better because of the tendency of TE to route along longer than shortest paths, under low loads. Fig. 4(b)-4(d) show the comparison of delays of individual flows for both schemes under three different load conditions. The x-axis in this case denotes flow IDs and the y-axis the average packet delays for the flows. Note that to remove clutter and the plots clearer, the flows are first sorted in ascending order of delays of single-path routing and then plotted. As can be seen, the proposed TE scheme significantly outperforms the single-path routing as the load in the network increases. In

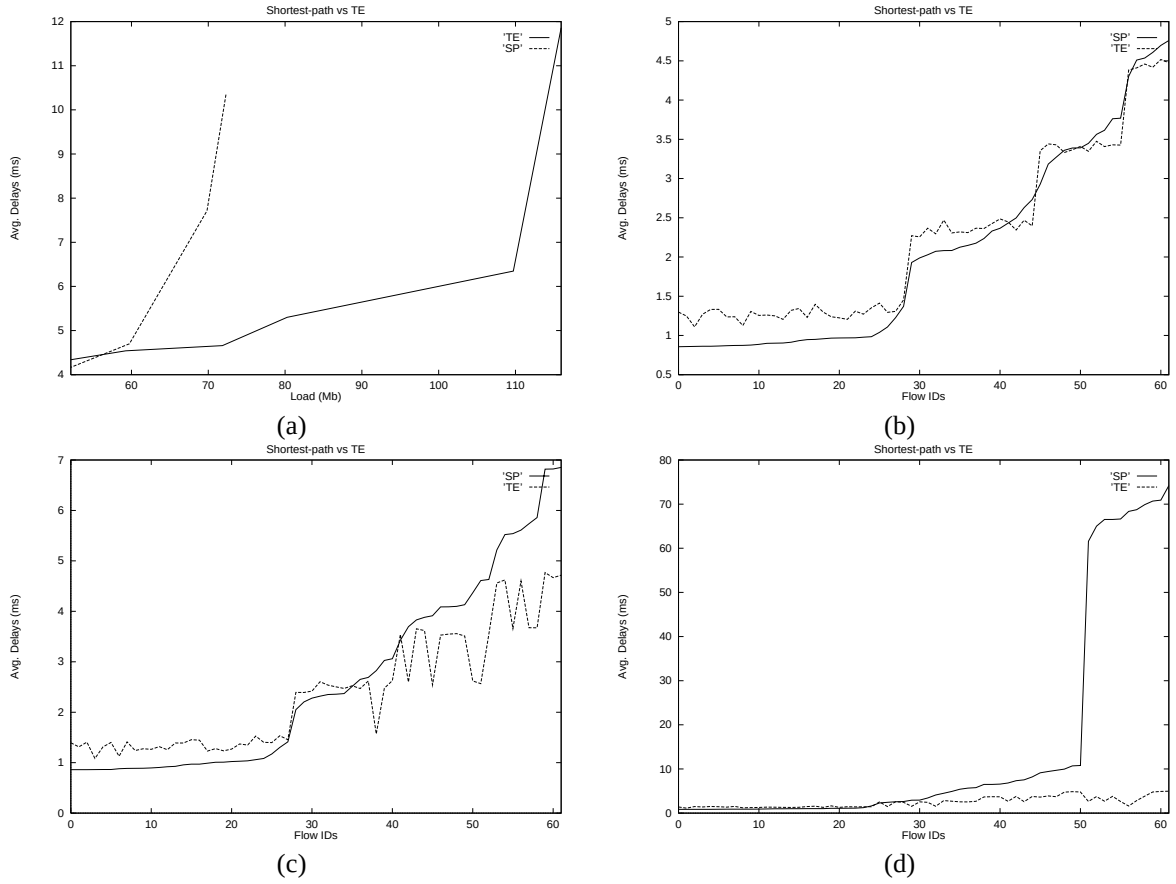

Fig. 4. Comparison of SPF and TE scheme

Fig. 4(b), for low loads, they are very close and often SP is better. Under high loads there can severe congestion in single-path routing which can be avoided in multipath routing as seen in Fig. 4(d). Because of the use of multipaths in the TE scheme congestion and, therefore, the delays are reduced.

Experiment 2: We test our assumption that when large number of TCP flows are present, delays close to minimum-delays can be obtained. TCP-10 indicates there are 10 flows between each source-destination pair. Curve UDP represents the optimal routing. Observe that in Fig. 5 as the total number of flows increases from 90 to 900, the delays decrease to levels comparable to the optimal. This is because of the finer granularity in distribution.

Experiment 3: Traffic can be forwarded more accurately according to routing parameters, if the presence of UDP traffic and packet sizes are considered during packet forwarding. This experiment tests this effect. We run the experiment for traffic that is split 60-40 between TCP and UDP. As the fraction of UDP traffic increases, the average delays improve (UDP-40). As mentioned earlier, this is due to reducing the skew introduced by TCP packet forwarding which OSPF-OMP does not do.

V. Conclusions

We proposed a traffic engineering (TE) system based on minimum-delay routing. For a given traffic matrix, multiple flows between each source-destination pair are determined using an offline algorithm and are then established using routing parameters. Packets are then forwarded according to routing parameters using hash techniques similar to OSPF-OMP. Our approach differs from OSPF-OMP in several significant ways. Firstly, we use IETF's Differential Services model to implement the TE system. The hash computation for a packet is done once at the ingress node and inserted into the packet. Within the network the hash key is directly used to determine the next hop without further hash computation. To further speed up forwarding the next-hop structure uses a table with next-hop entries instead of boundary values as in OSPF-OMP. Unlike OSPF-OMP, our system treats datagram and TCP traffic differently and takes into account the packet sizes giving finer granularity of traffic distribution. The use of Diffserv model enables complex operations, such as peeking into the TCP header, to be done only once at the ingress node. Also finer granularity of distribution can be achieved by decoupling hash computation from source-destination address.

Once problem that we have not addressed in this paper is adaptation to traffic fluctuations and adjacency link failures. We can adapt similar techniques in OSPF-OMP, but this problem is inherently difficult because our goal is to achieve minimum-delay routing rather than mere adjustment to congestion. Also the TE is described as applied to an autonomous system and does not address inter-domain routing. This is topic for further research.

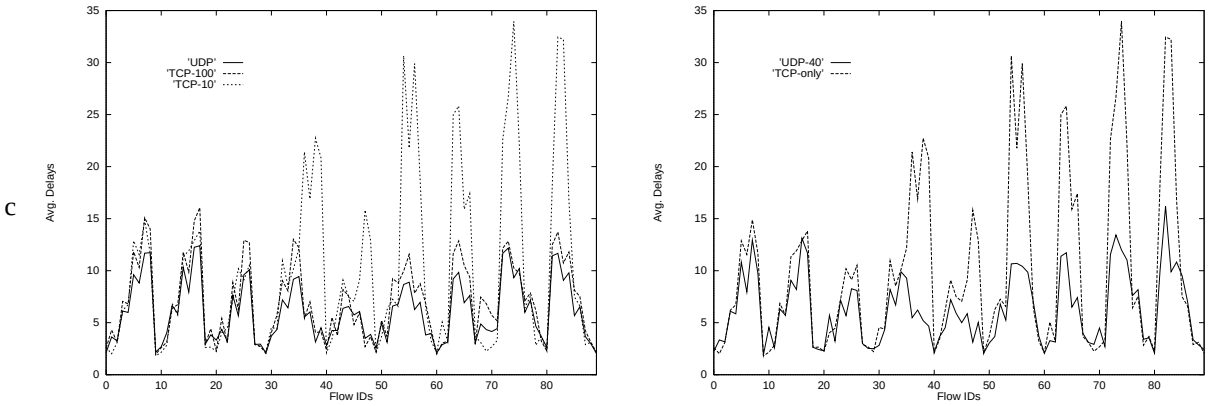

Fig. 5. (a)Delays as a function of volume of flows (b)Delays in presence of hybrid traffic

Our packet forwarding mechanism is quite general and can be used for other optimization criteria as in [23]. The system is practical and can be implemented in conjuction with the emerging Differential Services architecture. It offers significant benefits in terms of delay and throughput performance over single path routing.

REFERENCES

[1] R. Albrightson, J.J. Garcia-Luna-Aceves, and J. Boyle. EIGRP-A Fast Routing Protocol Based on Distance Vectors. *Proc. Networld/Interop 94*, May 1994.

[2] D. Awduche and et al. A Framework for Internet Traffic Engineering. *Internet Draft, URL:draft-ietf-tewg-framework-00.txt,* January 2000.

[3] D. Bersekas and R. Gallager. Second Derivative Algorithm for Minimum Delay Distributed Routing in Networks. *IEEE Trans. Commun.*, 32:911–919, 1984.

[4] D. Bersekas and R. Gallager. *Data Networks. 2nd Ed.* Prentice-Hall, 1992.

[5] D. Bertsekas and E. Gafni. Projected Network Methods and Optimization of Multicommodity Fflows. *IEEE Trans. Automatic Control*, 28:1090–1096, 1983.

[6] D. Katz et al. Traffic Engineering Extensions to OSPF. *¡draft-katz-yeung-ospf-traffic-01.txt¿,* Oct. 1999.

[7] Y. Bernet et al. An Framework for Differentiated Services. *Internet Draft,* May 1998.

[8] L. Fratt, M. Gerla, and L. Klienrock. The Flow Deviation Method: An Approach to Store and Forward Communication Network Design. *Netorks*, pages 97–133, 1973.

[9] R. G. Gallager. A Minimum Delay Routing Algorithm Using Distributed Computation. *IEEE Trans. Commun.*, 25:73–84, January 1977.

[10] C. Hendrick. Routing Information Protocol. *RFC*, 1058, june 1988.

[11] P. Humblet and S. Soloway. Algorithm for Data Communication Networks: Part I and II. *Codex Corporation,* 1986.

[12] W. Lai. Capacity engineering of ip-based networks with mpls. *Internet Draft, URL:draft-wlai-tewg-cap-eng-00.txt*, March 2000.

[13] P. M. Merlin and A. Segall. A Failsafe Distributed Routing Protocol. *IEEE Trans. Commun.*, 27:1280–1287, September 1979.

[14] J. Moy. OSPF Version 2. *RFC*, 2328, 1998.

[15] J. Moy. OSPF Version 2. *RFC*, 2328, 1998.

[16] A. Segall. The Modeling of Adaptive Routing in Data Communication Networks. *IEEE Trans. Commun.*, 25:85–95, January 1977.

[17] A. Segall and M. Sidi. A Failsafe Distributed Protocol for Minimum Delay Routing. *IEEE Trans. Commun.*, 29:689–695, May 1981.

[18] K. Takashima and et al. Concept of IP Traffic Engineering. *Internet Draft, URL:draft-takashima-te-concept-00.txt,* October 1999.

[19] D. Thaler. Multipath issues in unicast and multicast next-hop selection. *Internet Draft, URL:draft-thaler-multipath-05.txt*, Feb 2000.

[20] J. Tsitsiklis and D. Bertsekas. Distributed Asynchronous Optimal Routing in Data Networks. *IEEE Trans. Automatic Control*, 31:325–331, 1986.

[21] C. Villamizar. OSPF Optimized Multipath. *Internet Draft, URL:draft-ietf-ospf-omp-00.txt*, March 1998.

[22] S. Vutukury and J.J. Garcia-Luna-Aceves. A Simple Approximation to Minimum Delay Routing. *Proc. of ACM SIGCOMM*, Sept. 1999.

[23] Y. Wang and Z. Wang. Explicit Routing Algorithms for Internet Traffic Engineerin. *Proc. of ICCCN*, pages 582–588, 1999.

APPENDIX

A. Properties of Datagram Forwarding Algorithm

For a given routing parameters we have to show that the forwarding algorithm (Fig. 1) does not forward more than L bits to any of the next-hop, and hence the traffic distribution is fairly accurate for UDP traffic even on a small scale. And the maximum deficit is $(k-1)L$.

For simplicity, we slightly modify the notation as all distributors are the same. Assume a stream is divided into k streams according the routing parameters. Let W be the amount of traffic that arrived so far and W_k, the amount of traffic that is forwarded to stream k and ϕ_k be the corresponding routing parameter. Also W^p denotes the value of W when the p^{th} packet arrives. $A(t,\tau)$ is the amount of input traffic that arrives in the interval $[t,\tau]$ and let the amount of traffic that stream k receives in the same interval be $A(t,\tau)$. Note that at time t, W will be equl $A[0,t]$.

Theorem 1: In the algorithm in Fig.1 (a) for each substream k, $(K-1)L \leq A_k(t,\tau) \leq L + \phi_k \rho(t-\tau)$.

Proof: If a packet could not be scheduled by the algorithm, then $\forall k, W_k > \phi_k W$. This implies $\sum_k W_k > W$ which is impossible. This proves that every packet is successfully scheduled by the algorithm.

We first show $W_k \leq L + \phi_k W$. Let it be true upto processing of $p-1$ packets. Then, for all i, $W_k^{p-1} \leq L + \phi_k W^{p-1}$. Let the new packet p be assigned to queue k, we have $W_k^p \leq W_k^{p-1} + L$. This implies $W_k^p \leq L + \phi_k W^{p-1}$ because $W_k^{p-1} \leq \phi_k W^{p-1}$. Substituting $W^p = W^{p-1} + L$, we get $W_k^p \leq L + \phi_k W^p - L\phi_k$. Therefore $W_k^p \leq L + \phi_k W^p$. Because at initialization $W_k = W = 0$, from induction it follows $W_k \leq L + \phi_k W$. Because $W_k = A_k(0,t)$ and $W = A(0,t) \leq \rho t$, we have $A_k(0,t) \leq L + \phi_k \rho t$. It follows that $A_k(0,t) \leq \delta_1 + \phi_k \rho t$ and $A_k(0,\tau) \leq \delta_2 + \phi_k \rho \tau$ for $\delta_1, \delta_2 \in [0, L]$. Therefore, $A_k(t,\tau) \leq (\delta_1 - \delta_2) + \phi_k \rho(t-\tau)$ and $(\delta_1 - \delta_2) \leq L$. That $(K-1)L \leq A_k(t,\tau)$ directly follows. ■