Adaptive Multipath Routing for Dynamic Traffic Engineering

Ivan Gojmerac[1], Thomas Ziegler[1], Fabio Ricciato[2], Peter Reichl[1]

[1] Telecommunications Research Center Vienna (ftw.)
Donau-City-Str. 1, 1220 Vienna, Austria
{gojmerac,ziegler,reichl}@ftw.at

[2] INFO-COM Department, Univ. of Rome "La Sapienza"
Via Eudossiana 18, 00184 Rome, Italy
ricciato@coritel.it

Abstract−**This paper proposes Adaptive Multi-Path routing (AMP) as a simple algorithm for dynamic traffic engineering within autonomous systems. In contrast to related multipath routing proposals, AMP does not employ a global perspective of the network in each node. It restricts available information to a local scope, which opens the potential of reducing signaling overhead and memory consumption in routers. Having implemented AMP in ns-2, the algorithm is compared to standard routing strategies for a realistic simulation scenario. The results demonstrate the stability of AMP as well as the significant performance gains achieved.**

I. INTRODUCTION AND RELATED WORK

Efficient routing algorithms have always been among the core building blocks of any packet switching network. Whereas existing routing protocols are usually designed for achieving network robustness and fast network reconvergence in the case of failures, the research interest in the routing of Internet traffic has recently shifted towards traffic engineering, i.e., performance optimization of operational networks. This requires new approaches to old and well-known problems, as traditional routing protocols do not provide a straightforward methodology for optimal allocation of network resources with respect to given (and possibly changing) traffic demands. In this paper, we propose and investigate a simple adaptive routing algorithm which allows efficient load balancing without introducing any management overhead in the network.

The current Internet routing architecture is rather straightforward: within individual Internet domains, all links are assigned numerical values called *link costs*, which are used as a basis for the calculation of network paths. Paths between any two nodes in the domain are determined by minimizing the sum of link costs over all path candidates. The choice of paths can therefore only be influenced indirectly through appropriate setting of link costs.

On top of this architecture, routing protocols allow the dissemination of topology information throughout the network. There are two main types of intra-domain routing protocols with respect to the underlying path calculation algorithms: link-state protocols like OSPF (Open Shortest Path First [1]) are based on the Dijkstra algorithm [2] and require a global view of the network topology in every node, whereas for distance-vector protocols according to [3, 4], the local perspective is sufficient.

In most operational Internet Service Provider (ISP) networks, the link cost values are usually kept static for several hours or days, during which traffic always takes the same path from source to sink. In principle, this routing architecture in combination with standard routing protocols works fine, except for the case of network congestion, where traffic will be persistently routed also over congested links, even if parallel uncongested paths exist. This may lead to a significant reduction of network efficiency.

In order to solve this problem, engineers have developed many approaches. One possibility is the global optimization of link costs for a given traffic matrix [5]. Without introducing new technologies, this method enables performance gains in operational networks, but unfortunately the optimization problem is NP-hard and thus heuristic methods must be employed − not to mention the difficulties in obtaining realistic traffic matrices of a live network with elastic TCP traffic [6].

Multiprotocol Label Switching (MPLS) [7] is basically a technology for establishing label switched paths (LSPs), i.e., virtual circuits, between any pair of routers in a network domain. Thus, MPLS offers high flexibility in the choice of paths compared to classical IP routing and is therefore very useful for establishing virtual private networks (VPNs), but on the other hand introduces extensive network management requirements for allocating network resources to individual paths, path maintenance and the necessary adaptation of network configuration to current traffic conditions.

In contrast to the above mentioned solutions, which are usually centrally managed, decentralized approaches have also been proposed, e.g., the adaptation of routing to the current load conditions in the network as attempted in the early ARPAnet, where link costs were dynamically adjusted in proportion to the instantaneous link delays. This scheme performs well under low and medium load, but leads to link load oscillations for high load, due to the coarse granularity of traffic shifts, as multiple paths may be affected at the same time by a single link cost change [8].

Finally, the OSPF Optimized Multipath (OSPF-OMP) protocol aims at achieving optimal load distribution automatically and dynamically, using a link-state protocol flooding mechanism for informing all routers in the network about the load of every link. Based on this information, the routers can shift traffic from congested to less congested paths, and thus minimize the maximum link utilization in the network [9, 10]. However, OMP has two drawbacks we believe to be significant: firstly, it requires for every node sophisticated and memory consuming data structures for storing detailed information about all paths (i.e., link vectors) between the node and every possible destination. Secondly, the signaling overhead caused by the generation of new link load information messages is non-deterministic and therefore unpredictable.

In contrast to OMP, our Adaptive Multi-Path (AMP) algorithm proposal requires only simple data structures and produces deterministic signaling traffic overhead. AMP is based on providing nodes only with local network information about the links to its direct neighbors, whereas the dissemination of congestion information uses the recursive structure of periodical so-called *backpressure messages* (BMs). AMP is completely protocol independent and thus can easily be integrated in both link-state and distance-vector routing protocols, while nevertheless enabling the continuous and autonomous adaptation of routing to current network load conditions.

The remainder of the paper is structured as follows: Section II introduces in detail the elements of the AMP algorithm, i.e., the calculation of multipaths and link load metrics, the signaling and packet forwarding solution, and the load balancing mechanism. Section III discusses AMP parametrization, before Section IV describes our evaluation methodology, including ns-2 implementation aspects, topology, traffic model, and simulation scenario. Section V presents a number of simulation results on stability and performance of AMP, and Section VI concludes the paper with summarizing remarks and an outlook on current and future research.

II. THE ADAPTIVE MULTIPATH ROUTING ALGORITHM

Consider an arbitrary node X having a set Ω_X of neighbor nodes $Y_i \in \Omega_X$, $i \geq 0$, and let $z_i = \overline{XY_i}$ denote the generic directed link from node X to node Y_i. As already mentioned, OMP requires X to have global knowledge about all link loads in the network. This information is used for load balancing decisions between the different paths from X to all other nodes in the domain. In order to keep nodes up to date, link load information is propagated between all nodes and repeatedly refreshed by a message flooding mechanism which is triggered either by the time elapsed since the last update or the amount of load change in the last measurement.

In order to prevent the resulting unpredictable signaling overhead and avoid the extensive memory consumption of OMP, the AMP algorithm reduces X's network view from a global to a local one. With AMP, an increase in utilization on link z_i does not result in a multitude of other nodes reacting immediately to this change by off-loading some of their paths containing link z_i. In contrast, only X as end node of z_i is concerned and tries to shift traffic away from link z_i to feasible alternative paths as much as possible. Additionally, X sends BMs to each neighbor node Y_j, $j \neq i$, notifying it to which extent traffic coming from Y_j contributes to the congestion situation on link z_i. Fig. 1 depicts as an example a BM sent from X to Y_0, regarding congestion on links z_1, z_2, z_3... Recursively, Y_0 in turn passes on a similar congestion notification to its neighbor nodes, again in proportion to the extent of those nodes' respective contribution to congestion, etc. As BMs can be kept extremely small, it is possible to send them out periodically and thus avoid the signaling bursts typical for OMP.

The next subsections contain our proposal in detail. After describing the calculation of multipaths, we consider link load metrics for TCP traffic and present our signaling, packet forwarding, and load balancing mechanisms.

A. Calculation of Multiple Paths

Interior gateway routing protocols like OSPF [1] exclusively use the shortest paths towards the destination nodes, i.e., only paths with minimal cost. If there are multiple paths with the same minimal cost, traffic can be equally split among these so-called "equal cost multipaths" (ECMP).

As our goal is to reach near-optimal load balancing in our network, it is crucial to have as many paths available as possible. While therefore lifting the above mentioned restriction to using only best paths, at the same time we wish to avoid complicated MPLS-like or source routing path establishment and thus keep the path calculation as part of a simple distributed algorithm like, e.g., Dijkstra's [2]. Therefore, we extend the number of available paths by employing the *relaxed best path criterion* proposed by [9]. Here, any neighbor node which is closer in terms of cost to the destination than the current node is considered a viable next hop for multipath routing.

Note as an immediate conclusion that the relaxed best path criterion prevents the formation of routing loops, as the cost to the destination node strictly decreases with every hop along a path. Moreover, it can easily be integrated into the software of Internet routers, as only minor modifications of the existing algorithms are needed.

B. Link Load Metrics

Load balancing algorithms require a metric for link load in order to perform comparisons for deciding about traffic shifting. Simple fractional load, e.g., link utilization ρ defined as

$$\rho = \frac{\text{Carried Traffic Volume}}{\text{Link Capacity} \times \text{Time Period}}, \qquad (1)$$

works well if the compared load levels are low or medium. For high load levels with elastic traffic, e.g., TCP, many links may display a link utilization of close to 100%. Therefore, we use a more general metric, i.e., the *equivalent load*, ρ', which reflects the dynamic behavior of traffic passing through the link [9]. ρ' is defined as

$$\rho' = \max\{\rho, \rho \times K \times \sqrt{P}\}, \qquad (2)$$

with packet loss probability P. For TCP traffic, ρ' is related to the rate a TCP flow could have if no packet losses were experienced, based on the fact that TCP slows down roughly in proportion to the inverse of the square root of packet loss on the bottleneck link [11]. Therefore, ρ' is especially well-suited for comparing TCP load on different links.

In (2), the scaling factor K determines the packet loss boundary at which the ρ' value exceeds the link utilization, as for $K > 1 / \sqrt{P}$, the system reacts sensitively to packet losses. In our simulations, we set the value of K to 10, meaning that the equivalent load value ρ' will exceed the link utilization value ρ for packet loss probabilities greater than 1%.

C. Signaling

As already mentioned above, an essential part of AMP is the signaling mechanism employed. Let us consider link $\overline{Y_0X}$ as an example for describing the information required by Y_0 for each of its output links. Y_0 essentially requires two types of information: firstly, the equivalent load on the link $\overline{Y_0X}$, and secondly, information about the extent to which traffic routed from Y_0 via X contributes to congestion on the links $\overline{XY_i}$, $i \geq 1$, as well as the links in the network further downstream. As Y_0 can directly measure and calculate ρ' for link $\overline{Y_0X}$ itself, the remaining information required by Y_0 has to be obtained through the already mentioned mechanism of back-pressure messages (BMs).

$BM(X, Y_0)$, i.e., the BM sent from node X to node Y_0, should contain both directly measurable information about the explicit load situation on links $\overline{XY_i}$, $i \geq 1$, as well as indirectly obtained information about the situation further downstream as reported to node X by nodes Y_i, i = 1,2,...n, where n is the number of output links for node X. In order to keep the BM small, these 2n parameters are eventually mapped to a scalar $B_{X \to Y_0}$ describing the congestion situation on the downstream links of node X. For simplicity reasons, from now on we identify the generic $BM(X, Y_0)$ with its respective scalar content $B_{X \to Y_0}$. Then, $B_{X \to Y_0}$ may be considered as a function f of the mentioned 2n parameters:

$$B_{X \to Y_0} = f(\rho'_{\overline{XY_1}} ..., \rho'_{\overline{XY_n}}, B_{Y_1 \to X}, ..., B_{Y_n \to X}). \qquad (3)$$

To describe f, we first use the independence of output links and reduce the number of parameters to one per link by summarizing the situation on each link $\overline{XY_i}$ with a function g, i.e.,

$$g_i = g(\rho'_{\overline{XY_i}}, B_{Y_i \to X}) \qquad \forall i = 1, \ldots, n. \qquad (4)$$

As neither the output link nor the network beyond should be overloaded, the maximum function is a good candidate for g. This leads to the interpretation of g_i as "*effective equivalent load*" ρ'' for link $\overline{XY_i}$:

$$\rho''_{\overline{XY_i}} = \max\left\{ \rho'_{\overline{XY_i}}, B_{Y_i \to X} \right\}, \ i = 1, 2, \ldots, n. \qquad (5)$$

Coming back to the calculation of $B_{X \to Y_0}$, we further refine (3) by summarizing the n parameters $g_i \equiv \rho''_{\overline{XY_i}}$ according to a function h:

$$\begin{aligned} B_{X \to Y_0} &= f(\rho'_{\overline{XY_1}}, B_{Y_1 \to X}; \ldots; \rho'_{\overline{XY_n}}, B_{Y_n \to X}) \\ &= h(g(\rho'_{\overline{XY_1}}, B_{Y_1 \to X}), \ldots, g(\rho'_{\overline{XY_n}}, B_{Y_n \to X})) \\ &= h(\rho''_{\overline{XY_1}}, \ldots, \rho''_{\overline{XY_n}}) \qquad (6) \end{aligned}$$

We propose to define h as a weighted sum of the g_i, where the weight for link $\overline{XY_i}$ corresponds to the ratio between traffic on link $\overline{XY_i}$ that has arrived from node Y_0 via X and the total traffic on link $\overline{XY_i}$. This sum compresses all information which is available to node X about Y_0's contribution to the congestion situation on the downstream part of the network (the "*explicit congestion indication*" as defined in [12]):

$$B_{X \to Y_0} = \sum_{Y_i \in \Omega_X \backslash Y_0} \frac{\beta_{\overline{XY_i}}(Y_0)}{\beta_{\overline{XY_i}}} \cdot \rho''_{\overline{XY_i}}. \qquad (7)$$

Remember that Ω_X is the set of all neighbor nodes of node X, the downstream link between nodes X and Y_i is called $\overline{XY_i}$, $\beta_{\overline{XY_i}}(Y_0)$ is the number of bytes sent from node Y_0 via X to Y_i. Finally, $\beta_{\overline{XY_i}}$ denotes the total number of bytes sent from any node $\in \Omega_X \backslash Y_i$ via X to Y_i.

Note that calculating $\beta_{\overline{XY_i}}(Y_0)$ in (7) requires node X to map precisely traffic on the input link $\overline{Y_0 X}$ to the output links $\overline{XY_i}$, $i = 1, 2, \ldots, n$. This mapping is described in a so-called "In/Out Matrix" stored in node X (see Fig. 1). This
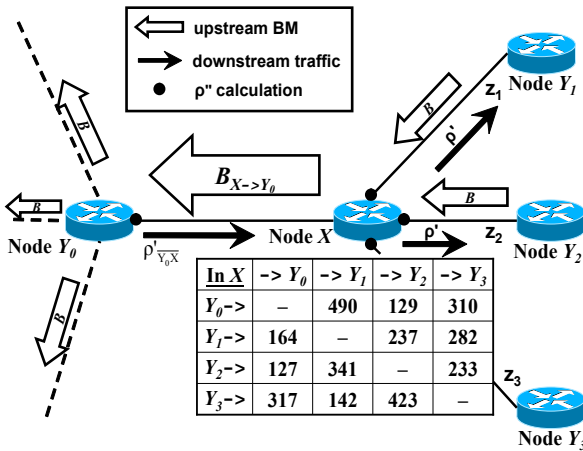
matrix contains for every node pair (P,Q), $P, Q \in \Omega_X, P \neq Q$, the number of bytes carried between these nodes via X.

D. Packet Forwarding

If an AMP node P has multiple paths towards a destination node Q, it must have a mechanism for splitting traffic destined to node Q among these paths. In contrast to ECMP, AMP enforces a mechanism which allows unequal splitting of traffic among the paths. The fundamental idea is to apply a hash function over the source and destination addresses, and to divide the hash space among the available paths by setting appropriate thresholds. Following [13] we chose the CRC-16 (16-bit Cyclic Redundancy Check) hash function as it achieves very good load balancing performance due to evenly spreading the source/destination address pairs in the solution space for most examined realistic traffic traces.

Whereas CRC-16 based implementations of ECMP divide the 16-bit (= 65,536 slots) hash space uniformly among the available next hops, this is not the case with AMP. Instead, in analogy to [9], the shares of hash space allocated to individual next hops are determined in proportion to the amount of traffic which should be sent out on the respective links as a result of the load balancing mechanism, which will be described in the following Section E.

E. Load Balancing

Whereas the signaling mechanism described in Section C is completely different from OMP, for load balancing we currently use a modification of the algorithm proposed in [9]. In the AMP version, decisions about load balancing are based only on the local view of the node. Therefore, instead of shifting traffic away from *paths* containing the "critically loaded link", AMP considers available outgoing *links* for every destination in the domain and attempts to distribute the load such that the ρ'' values as of (5) are approximately equalized.

As an AMP-specific consequence, it is crucial that the load balancing algorithm resorts to dynamic, but rather conservative load shifting. Because the received BMs may contain distorted information if a congestion event is very distant from the node (in terms of hop count), the load on a link should not vary significantly on the time scale related to the propagation of backpressure information across the domain.

Our algorithm performs *per destination* load balancing. Each destination is associated a set of viable next hops with their corresponding portions of the 16-bit hash space. The essential mechanism of the load balancing algorithm is changing the relative portions of hash space among the individual viable next hops. In order to make sure that the load balancing algorithm performs load shifting both quickly and oscillation-free, an intelligent progressive mechanism for the regulation of load balancing dynamics is included which works as follows: Initially, the rate of load shifting (i.e., the size of hash space changes) is small. While the direction of load shifts persists, the rate of load shifting increases exponentially. Once the direction of load shifting changes, the rate is again reduced to a predefined minimum value. In this way, the algorithm reaches the equilibrium point fast and with a minimum of overshoots and oscillations.

III. PARAMETER DISCUSSION

A. Stability Issues

AMP is a distributed feedback mechanism, and therefore special attention should be paid to its stability. It is well known that the stability of any control mechanism depends on the

Fig. 1: Example for a Backpressure Message (BM) from X to Y_0 and In/Out Matrix in Node X

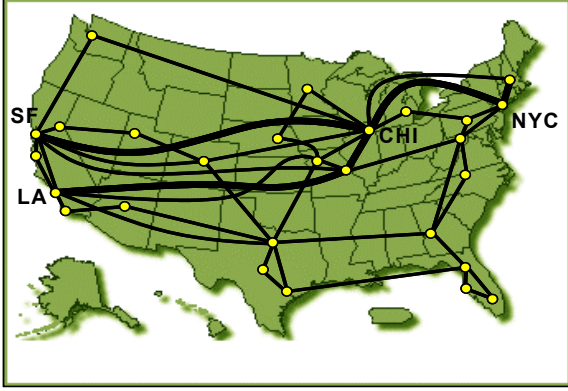| In X	-> Y_0	-> Y_1	-> Y_2	-> Y_3
Y_0->	–	490	129	310
Y_1->	164	–	237	282
Y_2->	127	341	–	233
Y_3->	317	142	423	–

Fig. 2: AT&T-US Topology

choice of control period and control law. In the case of AMP, we use an intelligent control law, as the control step size is chosen dynamically by the load balancing algorithm. Therefore, AMP's stability will ultimately be determined by the choice of the control period.

As any load shift at a generic node X may affect the loading of a distant downstream link w, it is beneficial for node X to receive information about possible distant congestion events before launching the next control step. Therefore, the control period should be proportional to both the backpressure interval τ_B, and the diameter of the network (i.e., maximum hop count D). We recommend that the control period τ_L, i.e., the interval between load balancing steps, be chosen as

$$\tau_L \cong D \times \tau_B. \tag{8}$$

This choice ensures that information about congestion even from the most distant links reaches any node approximately once between two consecutive control steps.

B. Signaling Overhead

We can analytically calculate the percentage of bandwidth required by the backpressure algorithm if we know τ_B, the size of the backpressure packet, and the link bandwidth. For instance in the case of OSPF, a minimal backpressure packet must contain the IP header, the OSPF packet header and the backpressure value. The IP packet header size is 20 octets, the OSPF packet header size is 24 octets, and we can generously assume that 4 octets are used for conveying the backpressure information. This gives us a total BM packet size of 48 octets. If we assume link capacity of 155 Mbit/s with $\tau_B = 1\,\mathrm{s}$, the AMP backpressure mechanism will consume $2.477 \cdot 10^{-4}\,\%$ of the link bandwidth. Therefore, we can conclude that the signaling overhead introduced by AMP is negligible even for very small values of τ_B.

IV. EVALUATION METHODOLOGY

A. ns-2 Implementation of AMP

For the purpose of performance evaluation, we have implemented the full functionality of the AMP algorithm in the ns-2 simulator [14], including the calculation of paths according to the relaxed-best path criterion, the calculation of AMP-specific link metrics, AMP signaling mechanisms, packet forwarding and load balancing. As far as the CRC-16 hashing is concerned, there is no fast software solution for ns-2. Moreover, ns-2 does not provide the required source and destination

IP addresses. Therefore, we have decided to assign each microflow upon its start a randomly generated integer ID. This solution is founded on the realistic assumption that CRC-16 is very effective in spreading microflows evenly in the 16-bit solution space [13]. Consequently, all packet forwarding decisions in the ns-2 nodes are based on comparing this random ID with appropriately set thresholds, thus significantly reducing the computational complexity, and making the simulation of large backbone networks with realistic traffic feasible.

B. Simulation Scenario

Using ns-2 packet simulations, we compare the performance of AMP with standard shortest path routing (SPR) and equal-cost multipath routing (ECMP). Our implementation differs from comparable related work on multipath routing performance evaluation [15, 16] which does not implement essential properties like the relaxed best path criterion, or uses unrealistically simple topologies and traffic models.

As far as a realistic topology is concerned, we have chosen the AT&T-US backbone topology [17] as sketched in Fig. 2. Here, except for five OC-192 links (i.e., 9.6 Gbit/s, bold lines in Fig. 2), the majority of network connections are OC-48, corresponding to a capacity of 2.4 Gbit/s. Link costs are set inversely proportional to the link capacities. For our scenarios, we have abstracted 27 core nodes and 47 backbone links from this topology.

For traffic generation, we use a state-of-the-art Web traffic model. Each Web user transmits a flow, stays idle for a user think time, and then transmits another flow, etc. Flow sizes and think times are Pareto distributed, according to the SURGE model as defined in [18]. In our simulations, each node hosts a virtual number of Web users proportional to the approximate population size of the corresponding US city where the node is located in reality. Similarly to [19], we model the spatial distribution (i.e., the fan-out) of HTTP requests in each node proportionally to the relative population sizes of the other nodes (i.e., cities) of the topology. Note that due to hardware limitations (the simulations have been performed on Pentium IV machines under SuSe LINUX), we had to reduce link capacities to 15 Mbit/s and 60 Mbit/s in our simulations.

Using the AT&T-US network topology, we have simulated the SPR, ECMP and AMP routing strategies under various load conditions. The different load levels were produced by linear scaling of the number of users at the individual nodes. We have computed the following different performance metrics:

- total TCP goodput,
- average response time for Web traffic,
- average coefficient of variation for the observed link utilization.

V. SIMULATION RESULTS

Figures 3 to 5 show the results of our ns-2 simulations, comparing SPR, ECMP and AMP. Note that in each case the initial simulation phase (before reaching equilibrium) has been dropped, and the remaining results correspond to a simulation time of 16000 seconds each. The various load situations are characterized by the total number of users in the system which is displayed at the x-axes.

Figures 3 and 4 represent the user perspective. As far as TCP goodput is concerned, AMP clearly outperforms the other two routing strategies (Fig. 3) by providing improvements of up to 28% in our simulation scenario. Note that the goodput
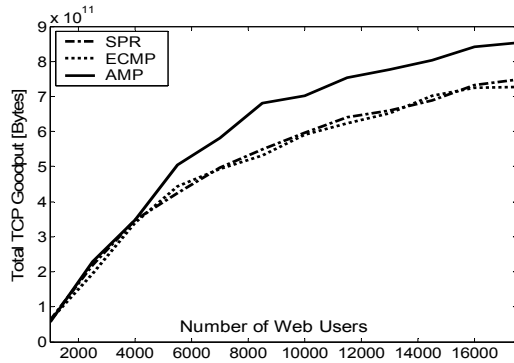
Fig. 3: Total TCP Goodput

Fig. 5: Average CoVs of Link Load

curve enters a state of saturation when the number of users approaches 17000, indicating that the system has reached the state of very high network load.

Even more impressive, Fig. 4 demonstrates AMP's benefit in terms of the most important user metric, i.e., the average Web page response time. In our simulations, the reduction in response time compared to SPR/ECMP reaches up to 43%, whereas both static schemes perform more or less identically.

Finally, Fig. 5 compares the average Coefficients of Variation (CoV) of link loads for the three routing schemes, which turn out to be quite close to each other. As SPR and ECMP by definition cannot display oscillations caused by adaptive routing, this fact proves the absence of oscillating behavior for the AMP algorithm and thus the stability of our algorithm throughout the investigated simulations.

VI. Conclusions and Future Developments

This paper deals with designing and evaluating Adaptive Multi-Path (AMP) as a simple routing algorithm for dynamic traffic engineering within autonomous systems. We have described the main novel features distinguishing AMP from related multipath routing proposals, especially the transition from global to local signaling and load balancing, thus causing significant reductions in signaling overhead, complexity, and memory consumption. After implementing AMP in ns-2, we have demonstrated both stability and performance of AMP compared to standard routing strategies for a realistic simulation scenario. Our performance evaluations show an improvement of up to 43% in terms of mean response times for Web traffic. Current and future work concerns the refinement of the AMP algorithm towards fast routing convergence in the presence of network faults.
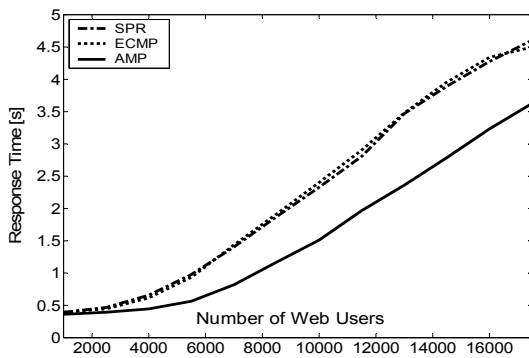

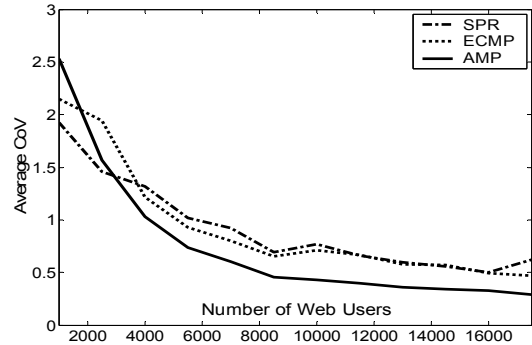Fig. 4: Average Web Page Response Times

References

[1] J. Moy: *OSPF version 2*. IETF RFC 2328, 1998.
[2] E. W. Dijkstra: *A note on two problems in connexion with graphs*. Numerische Mathematik, Vol. 1, 1959, pp. 269-271.
[3] R. E. Bellmann: *Dynamic programming*. Princeton, NJ, 1957.
[4] L.R. Ford, D.R. Fulkerson: *Flows in networks*. Princeton, NJ, 1962.
[5] B. Fortz, M. Thorup: *Internet traffic engineering by optimizing OSPF weights*. IEEE Infocom, Tel-Aviv, Israel, 2000, pp. 519-528.
[6] A. Medina, N. Taft, K. Salamatian, S. Bhattacharyya, C. Diot: *Traffic matrix estimation: existing techniques and new directions*. ACM SIGCOMM, Pittsburgh, PA, 2002.
[7] E. Rosen, A. Viswanathan, R. Callon: *Multiprotocol label switching architecture*. IETF RFC 3031, 2001.
[8] A. Khanna, J. Zinky: *The revised ARPANET routing metric*. ACM SIGCOMM Symposion on Communications Architectures and Protocols, Austin, TX, 1989, pp. 45-56.
[9] C. Villamizar: *OSPF optimized multipath (OSPF-OMP)*. IETF Internet Draft, 1999.
[10] C. Villamizar: *OMP simulations*. http://www.fictitious.org/omp/simulations.html .
[11] M. Mathis, J. Semke, J. Mahdavi, T. Ott: *The macroscopic behavior of the TCP congestion avoidance algorithm*. ACM Computer Communications Review, 27(3), 1997.
[12] I. Gojmerac, T. Ziegler, P. Reichl: *Adaptive multipath routing based on local distribution of link load information*. Proc. 4th COST 263 International Workshop on Quality of Future Internet Services (QoFIS'03), Stockholm, Sweden, October 2003.
[13] Z. Cao, Z. Wang, E. Zegura: *Performance of hashing-based schemes for Internet load balancing*. IEEE Infocom, Tel Aviv, Israel, 2000.
[14] ns-2 Homepage - http://www.isi.edu/nsnam/ns/ .
[15] K. Farkas: *OMP simulation results in IP networks*. PCH Conference, Budapest, Hungary, 2001.
[16] G. M. Schneider, T. Nemeth: *A simulation study of the OSPF-OMP routing algorithm*. J. Computer Networks, Vol. 39 (4) 457-468, 2002.
[17] E. J. Anderson, T. E. Anderson, S. D. Gribble, A. R. Karlin, D. J. Wetherall: *Towards efficient and robust adaptive routing*. Submitted to ACM SIGCOMM, Pittsburgh, PA, 2002.
[18] P. Barford, M. E. Crovella: *Generating representative Web workloads for network and server performance evaluation*. ACM Sigmetrics, Madison, WI, June 1998.
[19] S. Bhattacharyya, C. Diot, J. Jetcheva, N. Taft: *POP-level and access-link-level traffic dynamics in a tier-1 POP*. ACM SIGCOMM Internet Measurement Workshop, San Francisco, CA, 2001.