ELSEVIER

Computer Networks 39 (2002) 457–468

COMPUTER NETWORKS

www.elsevier.com/locate/comnet

A simulation study of the OSPF-OMP routing algorithm

G. Michael Schneider [a,*], Tamas Nemeth [b]

[a] Department of Mathematics and Computer Science, Macalester College, St. Paul, MN 55105, USA
[b] Deloitte Consulting, Minneapolis, MN 55402, USA

Received 20 September 2001; received in revised form 11 November 2001; accepted 6 February 2002

Responsible Editor: I. Nikolaidis

Abstract

Open shortest path first (OSPF) is the most widely used internal gateway routing protocol on the Internet. However, one shortcoming is that it does not take advantage of the existence of multiple equal-cost paths between source and destination nodes. A well-known variation of OSPF, OSPF-ECMP (ECMP, equal-cost multipath), does exploit the presence of multiple equal-cost paths, but only on a static basis. A variation of OSPF, OSPF-OMP (OMP, optimized multipath), attempts to dynamically determine the optimal allocation of traffic among multiple equal-cost paths based on the exchange of special traffic-load control messages. This paper briefly describes the OSPF-OMP algorithm and the design of a discrete event simulator that models its behavior. We then use this simulator to carry out three experiments that compare the performance of OSPF, OSPF-ECMP, and OSPF-OMP under a range of traffic loads and distributions. Our results show that OSPF-OMP produces improvements in both delivery time and the number of lost messages when compared with the other two protocols. © 2002 Published by Elsevier Science B.V.

1. Introduction

Open shortest path first (OSPF) is a link-state routing protocol developed by the Internet Engineering Task Force (IETF), and it is the internal gateway protocol currently recommended by the Internet Advisory Board [1,2]. Like any link-state protocol, it may identify a number of distinct equal-cost paths between source/destination pairs. However, unless the protocol has been explicitly configured to take advantage of these multiple paths, it arbitrarily chooses one route and uses it in all forwarding operations.

One of the earliest attempts at exploiting equal-cost routes was the OSPF variant called OSPF-ECMP, an acronym for equal-cost multipath [3]. ECMP divides the total volume of traffic across all equal-cost paths using fixed, unchanging measures such as line speed or hop count. Referring to the sample four-node network in Fig. 1, if our distance metric is hop count, then there are two equal-cost paths from A to D—ABD with cost 2 and ACD, also with cost 2. OSPF would arbitrarily choose one of these two paths and use it for all traffic arriving at A destined for D. On the other hand, ECMP would split traffic equally between these two routes. This splitting process could be done using *round-robin forwarding*, in which messages

[*] Corresponding author. Tel.: +1-651-6966-458; fax: +1-651-6966-518.

E-mail addresses: schneider@macalester.edu (G.M. Schneider), tnemeth@dc.com (T. Nemeth).

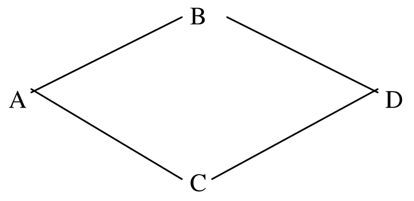


Fig. 1. Sample four-node network.

$1, 3, 5, \ldots$ go one way, while $2, 4, 6, \ldots$ go the other, or via *hashing* in which the source and destination host identifiers are hashed to produce a numerical value that determines the specific route taken.

However, since ECMP does not dynamically adjust its division of traffic based on a knowledge of current loads, its allocation may be sub-optimal. For example, referring again to Fig. 1, assume that all links have capacity 1, the traffic volume from A to D is 1.2 units, from B to D, 0.5 units, and from C to D, 0.2 units. Using OSPF, all A to D traffic travels via a single route. If that chosen route is ABD, then the BD link is utilized 170% ($1.2 + 0.5$ units of traffic, with a link capacity of 1 unit) while the CD link is utilized only 20%. If OSPF instead chooses the ACD path, then the BD and CD links will be used 50% and 140% respectively. In both cases links are overloaded, and the system is unstable. With OSPF-ECMP, traffic from A to D is divided evenly between the two equal-cost routes. This produces utilizations along the BD and CD lines of 110% and 80% respectively—more closely balanced, but still unstable. An optimal split, based on given loads, would allocate 35% of the A to D traffic to path ABD and 65% to ACD, resulting in a utilization of 85% on both BD and CD—a stable system. Unfortunately, since ECMP does not dynamically examine loading values, it is unable to determine this optimal split.

There have been previous attempts to create adaptive routing algorithms that respond to changing traffic loads and patterns. Unfortunately, they have generally met with limited success due to oscillation and protocol overhead [4]. Referring once more to Fig. 1, if ABD were the optimal route from A to D, then a dynamic protocol would shift most traffic to it, causing increased delays and longer queues along that route. This will eventually cause ACD to become the better route, which will cause longer delays and queues along it, causing us to switch back again to ABD, etc. Dynamic algorithms are extremely sensitive to the "granularity" of change. That is, if the protocol discovers a better route but switches too much traffic too quickly, that route becomes overloaded and sub-optimal, quickly leading to oscillation and instability. On the other hand, if too little traffic is switched, then we are not exploiting the adaptive nature of the protocol and will not gain much when compared to traditional static algorithms.

A second major problem with dynamic routing methods is *overhead*. If we update too often then nodes will always have current loading information but at the cost of excessive network overhead. If updates are infrequent, then the loading information is out-of-date, and decisions about how to distribute traffic may be quite poor.

OSPF-OMP, an acronym for "optimized multipath" is the most recent attempt to create a load-sensitive routing algorithm. It was proposed by Curtis Villamizar of UUNET Corp. in October, 1997 and initially presented to the IETF in March 1998 [5]. Modifications to the original proposal were made in both 1998 and 1999, and the most recent IETF draft is entitled *draft-ietf-ospf-omp-03*, dated August 18, 1999, and available on the Web at http://www.brookfield.ans.net/ospf-omp/. For additional information about any aspect of OSPF-OMP, check the OMP home page at http://www.brookfield.ans.net/omp/.

This paper gives an overview of the OSPF-OMP protocol and then describes a discrete event simulation model created by the authors to analyze the performance of the entire OSPF family of routing algorithms. It then presents the results of experiments conducted with this model to study the performance of OSPF, OSPF-ECMP, and OSPF-OMP under both normal and highly stressed traffic loads. Our observations are compared with the results of earlier simulation experiments on OSPF conducted by the IETF. Our results demonstrate the important performance improvements that can potentially be achieved using the OSPF-OMP adaptive routing protocol.

2. The OSPF-OMP routing algorithm

OSPF-OMP belongs to the general category of routing methods called *link-state routing algorithms* in which every node has a complete copy of the network map that is updated on a regular basis. Using this map each node executes a shortest-path first algorithm to determine the optimal routes. (For more information about the family of link-state routing methods, refer to [6,10].)

The three fundamental stages of the OSPF-OMP adaptive routing protocol are

- flooding of loading information,
- load adjustment,
- message forwarding.

We individually discuss each of these operations in the following three sections.

2.1. Flooding of loading information

Every network router samples its own SNMP counters at 15 s intervals and collects the following statistics about each link: (1) the fraction of link capacity used, called the *observed load*, (2) the number of incoming and outgoing packets dropped, and (3) the line speed in kilobytes/s. The observed load values are converted to a quantity called *equivalent load*, which is an estimate of what the loading would be if packets were not lost due to memory constraints and TCP window size restrictions. On lightly used lines the observed load and equivalent load will be virtually identical since packets are rarely lost. On heavily loaded lines, however, the equivalent load may be over 100% indicating that the total traffic exceeded a line's capacity. Thus, a link with an observed load of 0.98 (98%) may have an equivalent load of 1.20 (120%) when lost packets are factored in. The equivalent load computation is an attempt by OMP designers to obtain a more accurate metric of which links are most heavily used and should have some traffic offloaded onto other lines. (The exact rules for the computation of equivalent load are given in [7, pp. 6–7].)

Next, each node determines whether or not to flood the loading information just collected to all other nodes. As mentioned earlier, this is an important decision as too frequent flooding adds significantly to overhead while infrequent flooding leaves nodes with out-of-date information. This decision is based on three criteria: (1) elapsed time since the last flooding message, (2) the current values of equivalent load on all lines, and (3) the change in equivalent loading values since the last flooding message. Flooding can occur as often as every 30 s if a line is heavily loaded, and its load is changing rapidly. It can be as infrequent as every 20 min if a line is lightly loaded. The algorithm for determining whether or not to flood link-loading information is specified in [7, pp. 18–20]. If a node decides to send a flooding message, then the equivalent load data for all physically connected links is packed into a special LSA_OMP_LINK_LOAD routing control message and transmitted to every other router in the area [8].

2.2. Load adjustment

Each node that does routing maintains a *next-hop structure*. This is a list of all equal-cost paths from itself to every possible destination. In addition, each node identifies the *critical segment*—the one link in the network with the highest equivalent load—and determines whether or not each path in the next-hop structure does or does not include the critical segment. Finally, each node determines the highest equivalent load value along any single link of each path and calls this the *equivalent path load* for that path.

In Fig. 2, the same four-node network used in Fig. 1, assume that node A has received the equivalent load values shown in italics. These values were obtained from nodes B, C, and D via

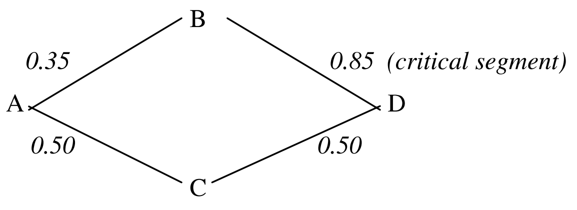


Fig. 2. Sample four-node network with equivalent loads.

Path	Contains Critical Segment	Equivalent Path Load	Probability
A → B → D	Yes	0.85	P_1
A → C → D	No	0.50	P_2

Fig. 3. Next-hop data structure for node A of Fig. 2.

the flooding technique described in the previous section.

Node A builds a next-hop structure containing all equal-cost shortest paths for reaching node D which, in this case, are ABD and ACD. The next-hop table for A to D routing might look something like the structure shown in Fig. 3.

The values P_1 and P_2 in Fig. 3 are the probability that a message at A destined for D would use this specific path. If we were using OSPF-ECMP both of these values would be initialized to 0.50 and would never change.

When a new LSA_OMP_LINK_LOAD routing control message arrives, it will trigger a load adjustment, i.e., a change to the probability values P_i, if (1) the message describes the equivalent load on the current critical segment—i.e., link BD in Fig. 2, (2) the message describes the traffic on a link whose equivalent load value will now make it the new critical segment, or (3) the difference in equivalent path load between the most heavily used path and the most lightly used path changes significantly.

For example, assume a LSA_OMP_LINK_LOAD message arrives at A from C saying that the load on link CD is 0.50. Nothing has changed so no traffic adjustments are made. Similarly, if the message reported that the load on CD had dropped to 0.40, no adjustments are made because the equivalent path load on path ACD is still 0.50. (The most heavily used link on this path is now AC, which has an equivalent load of 0.50.)

However, assume that the LSA_OMP_LINK_LOAD message reports the equivalent load on CD has increased to 0.60. Now the equivalent path load on ACD is 0.60, and criterion 3 above has been met. The difference between the most and least heavily used paths has changed from 0.35 (0.85–0.50) to 0.25 (0.85–0.60).

If the equivalent load on CD has grown to 0.90, that will also trigger a traffic adjustment due to criterion 2 above, since link CD becomes the critical segment. Finally, a LSA_OMP_LINK_LOAD message containing information about BD will trigger a traffic adjustment due to criterion 1, since BD is the current critical segment.

When a node adjusts its forwarding probabilities it decreases the probability value of all paths that include the critical segment and increases the probability value of all paths that do not contain the critical segment. The size of the adjustment varies; initially it is set to 1% but increases exponentially each time an adjustment is made in the same direction. This is done to help speed up the move toward equilibrium. However, whenever the direction of the increment is reversed, (i.e., the critical segment changes) the adjustment size is cut in half to reduce the likelihood that the algorithm will develop oscillations. While the stability of this algorithm has not been formally proven, initial IETF simulation studies strongly indicate a trend toward stable behavior [9,11].

A complete description of the OSPF-OMP traffic adjustment algorithm is contained in [7, pp. 11–12].

2.3. Message forwarding

Fig. 3 appears to imply that the information needed to divide traffic between equal-cost routes is represented as probabilities. While these values are probabilities, they are not actually stored in the "traditional" way—i.e., values in the range [0.0,1.0]. Instead, they are stored as *hashing boundaries*.

Nodes make routing decisions by hashing on source and destination IP addresses located in the packet header and using the result to determine how to forward the message. The OSPF-OMP hashing function produces a 16-bit unsigned value in the range [0,65535]. By dividing this range in a manner that reflects the probability of selecting each route, the output of the hash function can be directly related to the identity of the node where the message should be forwarded. In addition, the use of a 16-bit hash value allows for very fine adjustments to the traffic distribution.

For example, if two equal-cost routes, R1 and R2, were to be used equally, then we would simply set the hashing boundary to $65,536 \div 2 = 32,768$.

For a given (src, dst) pair within message M, if $H(\text{src}, \text{dst}) < 32,768$ then message M will take route R1, otherwise it will go via R2.

All paths in the next-hop table that have the same next node are folded together into a single entry in the routing table and assigned a single hashing boundary value. It is these hashing boundaries, not probabilities, that are used in making forwarding decisions. For example, assume the four-node network shown in Fig. 1 but with a bi-directional link between nodes B and C. Now, there are four paths from A to D and assume that the probability of using each of those routes is as follows:

Path	Probability
$A \to B \to D$	0.3
$A \to B \to C \to D$	0.05
$A \to C \to D$	0.5
$A \to C \to B \to D$	0.15

Since the first two paths both forward messages to node D via node B, we can fold these two paths together in the routing table and say that there is a 35% chance of forwarding a message to node B. (It is B that will ultimately decide whether to choose path ABD or ABCD.) Similarly, the next two paths can also be folded together, producing a 65% chance of forwarding a message to node C. The simplified routing table now looks like the following:

Destination	Next node	Hashing boundary
D	B	22,938(= 0.35 × 65,536)
	C	65,536

When A receives a packet destined for D, it first evaluates the function $\text{val} = H(\text{src}, \text{dst})$. If $\text{val} < 22,938$ the message is sent to node B; otherwise it is forwarded to node C.

We can now see how load adjustments fit quite naturally within this model. For example, based on newly arrived loading information, we may wish to decrease traffic along all paths that include the critical segment BD, and increase traffic along the segments that do not include the segment BD. This might produce something like the following set of modified probabilities:

Path	Probability
$A \to B \to D$	0.295
$A \to B \to C \to D$	0.051 (probability of sending to B = 0.346, previously 0.35)
$A \to C \to D$	0.506
$A \to C \to B \to D$	0.148 (probability of forwarding to C = 0.654, previously 0.65)

To reflect these new probabilities, we only need to adjust the hashing boundaries that are stored in the routing table:

Destination	Next node	Hashing boundary
D	B	22,675(= 0.346 × 65,536, previously 22,938)
	C	65,536

This has been only a brief description of the highly complex OSPF-OMP dynamic routing protocol. There are a large number of user-configurable options and parameters that allow a network manager to customize the specific behavior of the protocol. For a complete description of the OSPF-OMP dynamic routing protocol, the interested reader is referred to [7,9].

3. The simulator

The authors designed and built a packet-based, discrete event simulator capable of modeling the behavior of autonomous systems running either "pure" OSPF or one of the two variants described in Section 2: OSPF-ECMP and OSPF-OMP. A configuration preprocessor allows users to describe the exact network they want to simulate, including the number of nodes, processing speed, the speed and location of communication links, the amount of memory, the arrival rate, size and destination of packets, and the particular OSPF routing variant to be used.

3.1. The two-stage queueing model

Each node in the network is modeled as a two-stage queueing system as shown in Fig. 4. The first stage is the *processing stage*, which accepts packets, determines if they have arrived at their destination and, if not, moves them on to the second stage. The second stage is the *transmission stage*, and it models packet transmission from one node to another. Following transmission, the packet will be located at stage one of the next node. Thus, the life of a packet is modeled as a series of processing/transmission operations at each node on the path from source to destination.

Newly generated packets, as well as in-transit packets arriving from other nodes, are either stored in the stage 1 processor queue or, if the processor is idle, sent directly to the processor. There are two packet types in the model—*data packets*, which are characterized by their source, destination, and length, and *control packets*, characterized by their type. Currently, the only control message included in the simulator is the LSA_OMP_LINK_LOAD flooding message used by OSPF-OMP.

Processor queues have a finite capacity (set by the user), and if insufficient memory is available when a packet arrives, it is discarded and a lost-packet counter updated. Each node has a single processor that handles all packets in a constant amount of time. If a packet's ultimate destination is this node then the delivery time is recorded, and the packet is removed from the system. If it is

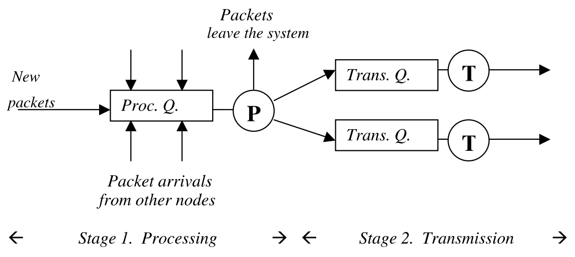


Fig. 4. Two-state queueing model of a network node.

destined for another node then a forwarding decision is made using the selected routing protocol and routing table, and the packet moves on to stage 2.

In the transmission stage there is a transmitter and queue for each outgoing line. The transmitter may start sending the packet immediately or, if it is busy, it puts the newly routed packet into the transmission queue of the corresponding line. The transmission time T of a packet is given by

$$T = (L + S/B)$$

where L is latency in s, B is line speed in bits/s, and S is the message size in bits. Transmission queues also have finite capacity and, when full, packets are again dropped and the lost-packet counter updated. If a packet is successfully transmitted it arrives at stage 1 of the next node on the path to its final destination, and this process/transmit sequence is repeated.

3.2. Traffic generation parameters

New messages are generated by the simulator at node i using an exponential interarrival time distribution with mean value λ_i, where λ_i is provided by the user during network configuration. Each node has its own local value of λ_i in order to provide the greatest amount of flexibility. In addition, the simulator contains an event type called change traffic parameters, which may be placed on the calendar as often as desired and for any time. When this event occurs, the simulator will input new values of λ_i from the configuration file and immediately begin using them.

The two other important traffic generation parameters are message size and message destination. The size of a message at node i is an exponentially distributed random variable with mean value λ_i. As with the generation rate, this mean value parameter is input by the user during network configuration and may be dynamically adjusted during simulation via the change traffic parameters event.

Finally, message destination is determined using a two-dimensional array called the destination probability table (DPT). The value of DPT$[i,j]$ is the probability that a message generated at node i

will be ultimately destined for node *j*. A random variable *r* is generated according to this distribution and used to determine the destination. The DPT table is provided by the user, and can also be dynamically adjusted during simulation.

4. The experiments

We ran a series of simulation experiments using the model in Section 3 to investigate the behavior of OSPF, OSPF-ECMP, and OSPF-OMP under a range of traffic intensities and distributions. All experiments were run using the 13 node, 16-link network shown in Fig. 5. This network roughly corresponds to the structure of the NSF Internet Backbone that existed in 1989 [10].

All communication links have the same speed, which translates to equal cost for all segments. As a result, of the possible $13 \times 12 = 156$ source–destination pairs, 30 have multiple equal-cost paths. This allowed us to test the ability of both ECMP and OMP to exploit the existence of these equal-cost routes.

4.1. Experiment #1: network performance under varying traffic intensities

In this first experiment we varied the amount of traffic in the network, and for each traffic level simulated network behavior for a period of 1 h. In all runs the number of new packets generated at each node is identical, and each node sends packets to all other nodes with an equal probability. We performed 20 experiments, with traffic rates at each of the 13-nodes scaled so that the total message generation rate for the entire network varied from a low of 100 mess/s to a high of 2000 mess/s in increments of 100. Each experiment was repeated five times, and the data reported in Table 1 is the average for these five repetitions. For each run we measured (1) the percent of packets dropped, and (2) the average end-to-end delivery time for all packets that reached their destination. We did this for the three variants of OSPF: best path always (BPA), or "pure" OSPF, ECMP and OMP. The results are displayed in Table 1, and they clearly indicate that for our specific network configuration OMP produced an improvement in performance at all traffic levels.

For light traffic loads, 100–500 mess/s, all three OSPF variants handle the total traffic without any lost packets. However, in every case OMP produced lower average packet delivery times compared to both BPA and ECMP. The improvements ranged from 2% to 9%, with an average reduction of 4.5%, compared to BPA, and from 1.5% to 8.6%, with an average of 4.2%, over ECMP. This shows that even with relatively light loads, the benefits of reducing queuing delays by making better use of equal-cost paths can be significant.

At medium levels of network load, 600–800 mess/s, both BPA and ECMP begin to lose packets due to queue overloads and memory limitations. However, OMP has no lost packets due to its ability to move traffic from heavily loaded lines to more lightly used ones and thus make better use of the total memory capacity of the network. At medium traffic volumes packet loss rates with BPA and ECMP are low—0.001–0.0055%—but that

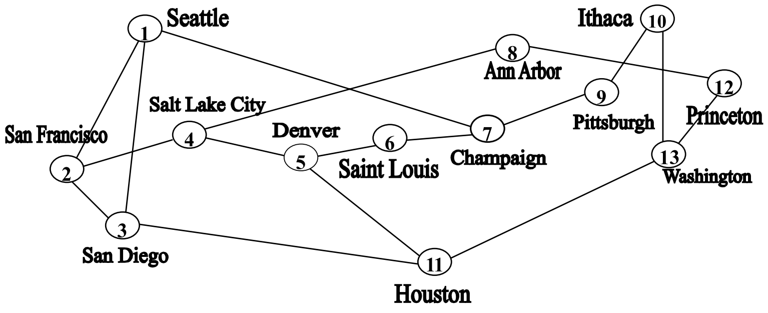


Fig. 5. The 13-node network used in the simulation experiments.

Table 1
Simulation results for Experiment #1

Mess/s	Average delivery time (ms)			Percent of packets loss		
	BPA	ECMP	OMP	BPA	ECMP	OMP
Light						
100	10.32	10.28	9.39	0	0	0
200	10.92	10.89	10.33	0	0	0
300	11.49	11.5	11.1	0	0	0
400	12.24	12.16	11.88	0	0	0
500	12.98	12.92	12.72	0	0	0
Medium						
600	13.85	13.75	13.53	0.001	0	0
700	14.94	14.76	14.57	0.0036	0.0021	0
800	16.27	15.94	15.78	0.00548	0.0049	0
High						
900	17.82	17.43	17.08	0.0202	0.0251	0.001
1000	20.92	19.57	18.76	0.128	0.0949	0.0117
1100	23.02	22.03	21.19	0.397	0.428	0.0352
1200	26.64	25.31	24.56	1.16	1.19	0.139
Saturated						
1300	30.64	28.83	29.85	2.52	2.54	0.384
1400	35	32.81	49.94	4.65	4.51	1.43
1500	39.5	36.78	64.91	7.28	6.52	3.84
1600	43.99	41.04	75.27	9.89	8.95	6.15
1700	48.63	45.28	82.8	12.9	11.5	8.6
1800	53.24	49.39	94.71	15.7	14.3	12.6
1900	57.07	53.65	100.1	18.5	17.2	15.9
2000	60.57	57.75	105.1	21.3	20.2	18

still represents a significant number of undelivered packets. For example, a packet loss rate of 0.00548% by BPA at 800 mess/s means that during the 1 h of simulated network behavior about 158 packets will be lost due to limitations of the routing protocol. In addition to not losing packets, OMP is also delivering packets faster than both BPA and ECMP, with an average improvement of 2.3–3% over BPA and 1.0–1.6% over ECMP.

It is at high traffic levels, 900–1200 mess/s, where OMP provides the most significant gains. While all three protocol variants are losing some information, OMP is doing by far the best job. With BPA and ECMP, the network infrastructure is unable to handle the load, and a relatively large percentage of packets are discarded—0.02% up to almost 1.2%. OMP loses between 0.001% and 0.14%. At a traffic rate of 1100 mess/s, this means that BPA and ECMP will drop about 16,000 packets during the 1 h of simulated behavior—a huge number. Under the same conditions, OMP would lose less than 1400. In addition to delivering more traffic, OMP is still delivering packets more quickly. At this load level there is a reduction in delivery time of about 7.5% over BPA and 3.2% over ECMP. (However, these numbers are highly variable because of the huge packet loss rate which effects the computation of the overall average delivery time.)

If we define a 0.1% packet loss as the maximum acceptable, then our simulation shows that both BPA and ECMP can handle traffic volumes up to about 1000 mess/s, while OMP can handle traffic loads up to 1200 mess/s. Put another way, with exactly the same set of hardware resources (links, processors, and memory) OMP can successfully handle a 20% increase in traffic intensity, a significant gain.

The one apparent anomaly in OMP performance is the average delivery time at traffic levels beyond 1300 mess/s—what we have termed a saturated network. A graph of average delivery time

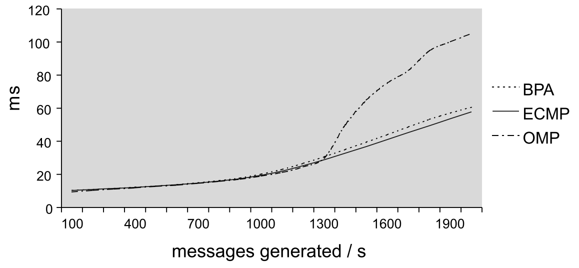


Fig. 6. Average delivery times for Experiment 1.

is shown in Fig. 6. Notice how the OMP values jump at 1300 and stay far above the other two lines for the remainder of the graph. The reason has to do with packet loss. At this level, both BPA and ECMP are losing enormous numbers of packets, as much as 21%. OMP is also losing packets, although at a significantly lower rate. Average delivery time measures the time of only those messages successfully delivered to their destination. At these saturated levels, so many messages are lost, that it is significantly affecting the computation. Messages that are lost by BPA and ECMP are being delivered by OMP, although it takes a long while, which increases average delivery time. In a sense, OMP is being "penalized" for doing a better job of delivering traffic during period of enormous load!

In summary, Experiment 1 demonstrated that:

- OMP performed better than both BPA and ECMP at light, medium, and heavy network loads. The improvements ranged from 2% to 7% over BPA to about 1–4% over ECMP.
- OMP did a significantly better job reducing the lost-packet rate. In a medium loaded network it did not lose any packets, while BPA and ECMP had packet loss rates of 0.001–0.005%. In a heavily loaded network it did lose packets, but at a far lower rate than either BPA or ECMP.

4.2. Experiment #2: response to changes in traffic patterns

Our second experiment examined how OMP would responds to changes in traffic distribution. Due to their static nature, both BPA and ECMP are incapable of adjusting their routing tables. OMP, on the other hand, can dynamically adjust traffic allocations based on changing conditions.

Using the same 13-node network in Fig. 5 we set the overall network load to 1000 mess/s (a high traffic rate according to Table 1) and ran the simulation for 60 s. In Experiment 1 all nodes sent an equal number of packets to all other sites. However, this type of totally uniform distribution is unrealistic, so at time 60 we changed the distribution. Now, only 5 of the 13-nodes generate new traffic, and packets are sent only to the other 8 nodes. This arrangement behaves somewhat like a client/server model—the five generators are similar to clients generating requests, while the eight receiving nodes are behaving like servers who must support a huge volume of incoming traffic.

We ran the simulation for another 900 s, and the results are shown in Fig. 7. The behaviors of both BPA and ECMP were quite similar, so we show only a single line for both.

Before the traffic change, OMP had an average delivery time of about 18–19 ms, while BPA and ECMP were delivering messages in about 21 ms. After the change in traffic pattern, delivery times rose for all three methods, reaching about 26 ms after 5 min. For ECMP and BPA, which are both static methods, delivery times remained at this level for the remainder of the simulation, an increase of about 19%.

OMP's behavior, however, was quite different. As with the static methods, delivery times increase immediately after the traffic change since the

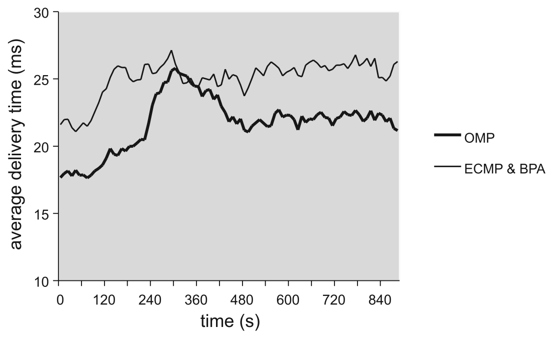


Fig. 7. Responsiveness to changes in traffic patterns.

protocol is using allocation parameters that are no longer appropriate for the new traffic distribution. The delivery times reach a maximum value of about 25 ms after 5 min of simulation. However, by this point in time nodes are beginning to note that some links are overloaded, and they start flooding LSA_OMP_LINK_LOAD messages which describe these new loads.

Upon receipt of these control messages nodes begin adjusting their hashing boundaries to redistribute the traffic, exactly as described in Section 2. It takes a while, since the early traffic adjustments are small, but eventually the network does adapt. After 8 min of simulation, and only 3 min after it reached its maximum value, OMP has reduced average delivery time to 21 ms. Delivery times remain stable at this level for the remainder of the simulation, without either instabilities or oscillations.

This is a clear demonstration of the ability of OMP to respond quickly and effectively to changes in the pattern of network traffic.

4.3. Experiment #3: behavior under unsteady state

In addition to changes in traffic distribution, another common characteristic of network traffic is massive, short-duration fluctuations in volume—i.e., an unsteady state. In order to study how the three variations of OSPF react to these capacity fluctuations we conducted an experiment where we suddenly doubled the traffic volume for a short period of time, namely 3 s, and then watched what happened with each of our three protocols. This behavior might simulate, for example, the transmission of a massive data file across the network. For this experiment we measured the total number of messages in the network, rather than delivery time. With high intensity bursts of traffic, you are usually more interested in how much traffic the network can carry than in the time it takes to deliver that traffic.

Using the network from Fig. 5, we started with the same simulation parameters as Experiment 1—a traffic intensity of 1000 mess/s and an equal distribution of packets to all other nodes. We ran the simulation until the network was stable and unchanging. Then we suddenly increased the traffic load from 1000 to 2000 mess/s for a period of 3 s. The results are shown in Fig. 8, which displays the total number of messages within the network against time relative to the instant when traffic intensity changed.

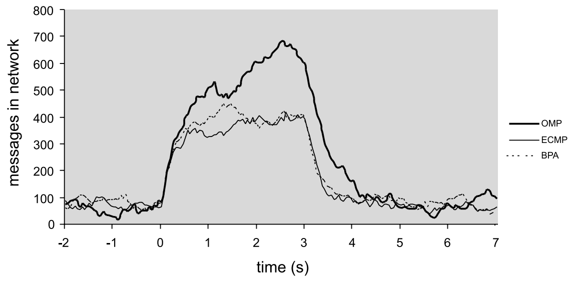


Fig. 8. Performance under traffic shock.

At time 0 traffic was doubled to 2000 mess/s, well beyond the capacity of the network as shown in Table 1. With both BPA and ECMP, the average number of messages in the network jumped from 100 to 400 in about one second and then stabilized.

The processor and transmission queues along the routes filled up quickly, and at that point the system was unable to handle any more incoming packets, and they were lost. It is possible that there was queue space available at other nodes, but neither BPA nor ECMP could adapt and make use of that free space. In a sense, Fig. 8 is saying that the "network capacity" of both BPA and ECMP is approximately 400 messages, and beyond that, all message traffic will be lost.

OMP, on the other hand, utilized the available memory resources much better. The number of messages in the system grew to about 700 messages in 2–3 s, an increase of 75% over what could be handled by the other two methods. That is, OMP was better able to accommodate these higher traffic levels due to a superior distribution of load across equal-cost paths, which made better use of the total amount of memory space available throughout the network.

At time 3, traffic intensity was reset to 1000 mess/s. BPA and ECMP returned to their original levels in about half a second. It took OMP longer, about 1–1.5 s, since it has more messages in the system that needed to be delivered.

4.4. Comparison with previous simulation work

The IETF has run simulations on the OMP routing protocol, and a summary of their results can be found in [11]. Their experiments differ from the ones described in this paper in that they were investigating the fundamental traffic allocation, stability, and convergence properties of the OMP algorithm described in Section 2. They examined how OSPF-ECMP and OSPF-OMP allocated load to equal-cost paths under different operating conditions, and compared that to the loading properties of basic OSPF. For example, in their initial experiment they used a 12-node, 19-link network (quite similar to Fig. 5) and a fixed traffic volume to study the link loading behavior of both ECMP and OMP. With ECMP they found that two of the network links were allocated 109% of capacity, an obviously unstable system. Running OSPF-OMP, they were able to reduce the load on the most heavily used link to 89%. Further experiments are described in [11].

Based on these simulation studies, the conclusions reached by the IETF regarding the relative performance of OMP vs. ECMP were:

- ECMP in some cases provides improvement over OSPF without equal-cost paths and in some cases makes loading worse. Having ECMP available as an option is clearly preferable to not having it available.
- OMP in all cases provided a loading improvement over OSPF without equal-cost paths and in all cases provided a loading improvement over ECMP. Simple topologies can be constructed where OMP performs no better than either of these two techniques, but OMP would never perform worse. In all real cases, OMP would provide better loading and in some cases very significantly better loading than OSPF without equal cost or with ECMP [11].

5. Conclusions

Our experiments with the network simulator have allowed us to get some preliminary estimates of the improvements in message delivery times and packet loss rates that are possible using the OSPF-OMP routing protocol in place of existing implementations of OSPF. Of course, the results presented in this paper apply only to the specific conditions that existed within our experiments, and these results cannot be generalized to statements about the overall performance that can be expected from OMP under all types of operating conditions.

However, based on the outcomes of the three experiments described in the previous section we observed that OMP performed better than either "pure" OSPF or OSPF-ECMP under a wide range of different traffic conditions. Specifically:

- OMP performed better than both BPA and ECMP at light, medium, and heavy network loads. The improvements in message delivery times ranged from 2% to 7% over BPA to about 1–4% over ECMP.
- OMP did a significantly better job in reducing the lost-packet rate. In a medium loaded network it did not lose any packets, while BPA and ECMP had packet loss rates of 0.001–0.005%. In a heavily loaded network it did lose packets, but at a far lower rate than either BPA or ECMP.
- OMP responded well to dramatic changes in traffic distribution. The response occurred in a short period of time and did not appear to suffer from oscillations.
- OMP responded well to dramatic fluctuations in traffic volume. It responded by making better use of the total amount memory available in the network to lose significantly fewer packets.

In the future we plan to use our discrete event simulator to study other aspects of OSPF-OMP and to examine how it behaves with different networks, a different set of lines, and different traffic characteristics.

References

[1] J. McQuillian, I. Richer, E. Rosen, The new routing algorithm for the ARPANET, IEEE Trans. Commun. 28 (5) (1980) 711–719.

[2] J. Moy, OSPF, Addison-Wesley, Reading, MA, 1998.

[3] J. Moy, OSPF, Version 2, IETF Technical Report RFC 2328, 1998. Available from <ftp://ftp.isi.edu/in-notes/rfc2328.txt>.

[4] S.H. Low, P. Varaiya, Dynamic behavior of a class of adaptive routing protocols, Proceedings of the Conference on Computer Communications, March 1993.

[5] Internet Engineering Task Force, Report RFC 1247, March 1998.

[6] L. Petersen, B. Davies, Computer Networks, second ed., Morgan Kaufman, Los Altos, CA, 2000, Section 4.2.3, Link state routing.

[7] C. Villamizar, Internet Engineering Task Force INTERNET-DRAFT, draft-ietf-ospf-omp-03, August 1999. Available from <http://www.fictitious.org/ospf-omp/ospf-omp.html>.

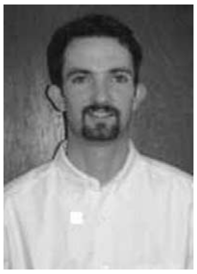
[8] R. Coltun, The OSPF opaque LSA option, Technical Report RFC 2370, Internet Engineering Task Force, 1998.

[9] Optimized Multipath Home Page. Available from <http://www.brookfield.ans.net/omp/#adjust>.

[10] A. Tanenbaum, Computer Networks, third ed., Prentice-Hall, Englewood Cliffs, NJ, 1996.

[11] http://www.brookfield.ans.net/omp/simulations.html.

G. Michael Schneider is Professor and Chair of the Department of Mathematics and Computer Science at Macalester College, St. Paul, Minnesota. He received his Ph.D. in Computer Science from the University of Wisconsin in 1974 and taught at the University of Minnesota for 8 years before moving to Macalester in 1982. His areas of research include computer networks, network protocols, and distributed systems. He is also the author of seven textbooks in computer science.



Tamas Nemeth was born in Kaposvar, Hungary. He received a Bachelor's degree in Computer Science, Mathematics and Physics from Macalester College in Saint Paul, Minnesota. He currently works at ORC Macro as a Senior Programmer Analyst. Among other honors, he received the first place prize from the Society of Industrial and Applied Mathematics and the Institute for Operations Research and Management Sciences in the International Mathematical Modeling Contest with his submission entitled “A Tricubic Interpolation Algorithm for MRI Image Cross-Sections”. His interests include ornithology and snow sculpting at the annual International Snow Sculpting Championships in Breckenridge, Colorado.