Shortest Path First with Emergency Exits

Zheng Wang

Jon Crowcroft

Department of Computer Science, University College London
London WC1 6BT, United Kingdom

Abstract — **Under heavy and dynamic traffic, the SPF routing algorithm often suffers from wild oscillation and severe congestion, and results in degradation of the network performance. In this paper, we present a new routing algorithm (SPF-EE) which attempts to eliminate the problems associated with the SPF algorithm by providing alternate paths as emergency exits. With the SPF-EE algorithm, traffic is routed along the shortest-paths under normal condition. However, in the presence of congestion and resource failures, the traffic can be dispersed temporarily to alternate paths without route re-computation. Simulation experiments show that the SPF-EE algorithm achieves grater throughput, higher responsiveness, better congestion control and fault tolerance, and substantially improves the performance of routing in a dynamic environment.**

1. INTRODUCTION

Shortest-Path routing algorithms commonly used in today's computer networks fall into two main classes: *distance-vector algorithm* and *link-state algorithm*. In a distance-vector algorithm, each node maintains a routing table containing the distance of the shortest path to every destination in the network. A node only informs its immediate neighbors of any distance changes to any particular destinations. In contrast, in a link-state algorithm, each node keeps track of the entire network topology and computes the routing table based on the link distance information broadcast by every node in the network.

In 1979, a version of the link-state algorithm, known as *Shortest-Path-First (SPF)*, was implemented in the ARPANET to replace the old distance-vector routing algorithm [McQU80][ROSEN80]. Operational experience has verified that the SPF algorithm responds to changes more quickly and does not suffer from long-term routing loops. It has in many ways improved the performance of routing in the ARPANET.

However, the introduction of high-speed networking and innovative real-time applications has made the network environment more dynamic. Some problems with the SPF algorithm have been recognized [WANG90][KHAN89][SEEG86][BERT82]. In particular, the SPF algorithm often exhibits instability and produces poor quality routes under heavy and dynamic traffic.

In this paper, we first discuss the problems associated with the SPF algorithm and then introduce a new routing algorithm which eliminates many of those problems and substantially improves routing in a dynamic environment.

2. PROBLEMS WITH THE SPF ALGORITHM

The SPF routing algorithm is based on the quasi-static model in which the link distance remains approximately constant during the route updating period. It serves remarkably well when the traffic load is light and changing slowly. Nevertheless, with the increase in the amount of the traffic and its burstiness, the quality of the routes often deteriorates. The SPF algorithm tends to become instable and often results in oscillation of routes, which in turn aggravates the congestion that the network may have been experiencing.

Many of the problems are associated with the distance estimation procedure of the algorithm and the characteristics of single path routing algorithms. We will only discuss briefly those problems in the following section. For a more general and detailed description, see [WANG90].

1) The *Maximum Flow* between two points represents the the dynamic range of traffic that the network is able to handle. According to the well-known *Max-Flow Min-Cut Theorem*, the maximum flow between any two arbitrary nodes in a network is equal to the capacity of the minimum cut separating those two nodes. The SPF algorithm, however, can achieve far less than the theoretic potential. The SPF algorithm is a single path routing algorithm, ie. at a given time, there is only one path for any source and destination pair. The maximum flow can only reach the capacity of bottleneck of the best

path between the pair. When the traffic is bursty, although the average load over a path may be low, the momentary traffic could exceed the capacity of the path several times and well result in loss of packets.

2) The SPF algorithm under heavy traffic often leads to oscillation. In the SPF algorithm, link distances such as delay, link utilization are measured as estimates for the future routing decisions. As the traffic approaches the capacity of the path, the link distance tends to rise sharply. The congested links often report very high distance value so that they may be abandoned by all traffic in the next route updating period and shift the congestion to other paths. At the next route updating point, those links will report a low distance and become congested again. When the traffic load approaches the capacity of the path, any attempt to search for a less congested path often results in wild oscillation and spread of congestion.

3) It is highly desirable that routing algorithms respond to traffic changes dynamically. However, in a large network with heavy traffic, high responsiveness can be difficult to achieve [SEEG86]. First, it often takes quite long time for routing information to travel across a large network. The routing algorithm can not respond to the network conditions at a rate faster than the rate at which relevant information can reach the points concerned and corresponding action can be taken. Second, route computation requires substantial amount of memory and CPU resources. Frequent route updating may affect the function of packet processing and forwarding. Third, routing updates are distributed across the network by flooding. Hence, each update flows at least once on each link. Frequent route updating may consume unacceptable bandwidth. Finally, the routing updates have higher priority than users' traffic. Transmission of large amount of such packets can have effects on the flow of users' traffic. Route updating in a large network is a slow and costly operation. Under heavy traffic load, it can lead to wild oscillation and degradation of performance. In practice, the frequency of route updating is often chosen to be quite low to ensure the stability of the network.

4) The SPF algorithm has, in principle, a built-in ability for congestion control. When a path is overloaded, the reported link distance increases. The routing algorithm re-computes the routing table and reduces the traffic over the congested path. Nevertheless, the amount of traffic to be shed from the congested link is difficult to predict as it largely depends on the composition of the traffic flow. When the traffic consists of many small flows over different source-destination pairs, appropriate amount of traffic can be shed by carefully turning the link metrics. But if the traffic is dominated by several large flows, re-computation of the routing table can not solve the problem and may lead to oscillation [KHAN89]. Congestion occurs when the traffic and resource mismatch at some points of the network. It is therefore usually local and lasts comparatively short time. Updating routing table, which involves exchanges of information and computation across the entire network, may not be appropriate. Further, when congestion does occur, the routing algorithm has to wait until next updating point to respond. At that time, the congestion may have already dissipated. The reported high link distance caused by the congestion, however, has misleading effects on next route updating.

5) Routing algorithms have to provide a certain degree of fault tolerance. When resource failures have been detected, routing algorithms have to respond quickly to re-calculate the routing tables and provide alternate paths so that the existing connections can survive. However, it is often very difficult to detect the failures within short time after they occur. In the SPF algorithm, the traffic may still be routed along the failed path until the failure is detected or the routing table is re-calculated. Thus the packets may accumulate in the network and lead to congestion and affect other users. Real-time applications are particularly vulnerable to those failures.

3. OVERVIEW OF THE NEW ROUTING ALGORITHM

A distributed routing algorithm can be decomposed into four procedures: distance measurement, information updating, route computation and packet forwarding. The distance measurement procedure monitors and collects certain network parameters according to the particular routing metric used. The collected information is distributed over the entire network by the information distribution procedure. In each node, the route computation procedure then constructs the routing table based on the received information and the packet forwarding procedure actually routes the traffic to the next hop. The new routing algorithm is an improvement over the SPF algorithm. It uses the same distance measurement procedure and information updating procedure as the SPF algorithm. Each node measures the actual delay to its neighbors and periodically broadcasts the changes to all other nodes in the network. However, the new routing algorithm is equipped with more sophisticated route computation procedure and packet forwarding procedure to deal with heavy and dynamic traffic.

The problem with the SPF algorithm is that there are no mechanisms to alter the routing other than updating the routing tables while route updating is too slow and costly for responding to traffic fluctuations. Under heavy traffic load, frequent route updating may also lead to instability. To solve this dilemma, the new routing algorithm maintains a stable routing table and meanwhile provides alternate paths to disperse traffic when it is accumulating in some points of the network. When congestion and network failures do occur, instead of initiating route updating, the node forwards the traffic along the alternate paths temporarily and pass around the congested or failed areas. If the changes are persistent, the routing tables will be updated eventually when the next route updating time is due.

In the new routing algorithm, the alternate paths are only used as *emergency exits* when the shortest paths are experiencing problems. In normal conditions, the new routing algorithm performs exactly the same as the SPF algorithm does. Because of this feature, we call the new routing algorithm *Shortest Path First with Emergency Exits (SPF-EE).*

It should be emphasized that the alternate path as an emergency exit does not have to be the shorest path to the particular destination, nor does it have to be disjoint from the current shortest path. When congestion or network failures occur, the primary goal is to find an alternative path for the traffic and avoid the packets accumulating in the network. The alternate paths provide a local and temporary adjustment of routing for the traffic to bypass the point or the area in question.

Fig.1 illustrates the basic concepts in the SPF-EE algorithm. Node x normally forwards packets for destination z to neighbor NS_z, where NS_z is the next-hop along the shortest path (SP) to the destination z. But if the queue length to node NS_z (denoted by Q_{NS_z}) exceeds a certain limit, the packet is transmitted to neighbor NA_z instead, where NA_z is the next-hop along the alternate path (AP).

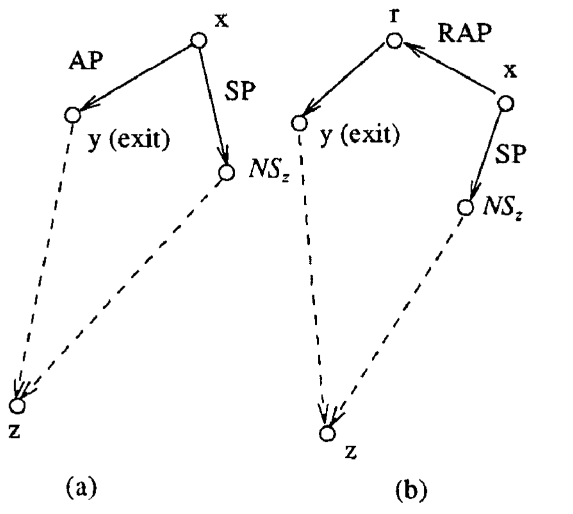
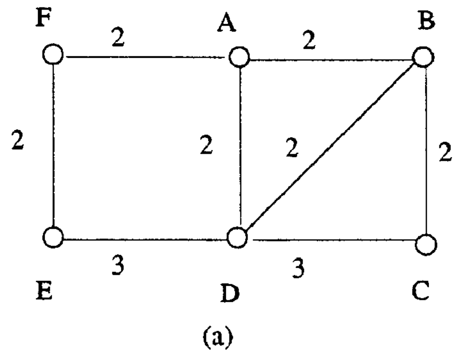


Fig.1: Alternate Path and Reverse Alternate Path

When a packet is forwarded to a neighbor y other than the next-hop of the SP, there are only two possibilities. If the neighbor y is not upstream from the node x in the sink tree for the destination z, the packet will travel along a SP from node y to the destination z (Fig.1(a)) without forming any loops. We call node y an *exit* for the destination z. However, if all the neighbors other than node NS_z are upstream from node x, the packet will be looped back to node x. In this case, node x sends a control packet to all its neighbors other than NS_z to inquire whether they have any exits for the destination z. Upon receiving the control packet, each neighbor checks to see whether it has a neighbor which is not upstream from node x. If an exit is found, it sends a reply back to node x and establishes a *reverse alternate path* (RAP). Otherwise, it propagates the control packet further to its neighbors until an exit is found. When using a RAP, the packet has to be source-routed to the exit (Fig.1(b)) and it then follows the SP to the destination. It is interesting to note that in some cases the packet may be sent backwards to the nodes it just comes from and then take a different path. APs can be viewed as a special case of RAPs. With APs the exits are next-hops while with RAPs the exits are more than one hops away.

In theory, the algorithm fails only when none of the upstream nodes have contact with other branchs of the sink tree. This occurs only when the network is partitioned.



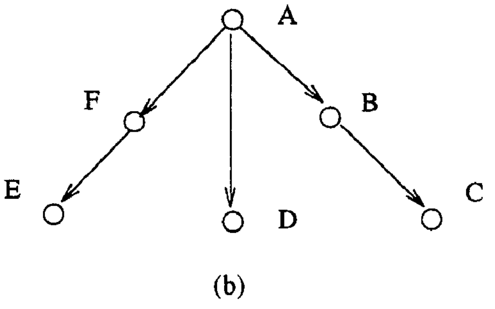


Fig.2: Network Topology and Routing Tree for Node A

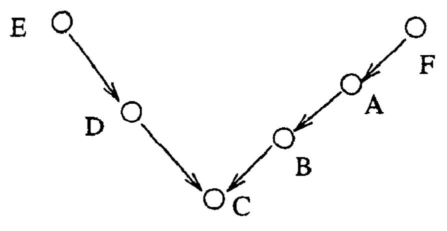
To illustrate the algorithm, consider a six-node network of Fig.2(a), with the routing tree for node A shown in Fig.2(b).

Traffic destined to node C is normally routed along the shortest path via node B. However, if for some reasons the link AB is congested, node A has to find an alternate path to node C. According to the sink tree for node C shown in Fig.3(a), node A has a neighbor node D which is not upstream from node A in the sink tree for node C. Thus an AP is available via node D.

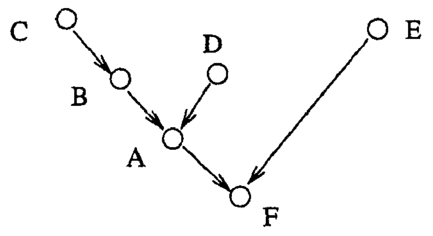
For destination F the situation is different as both neighbors (B and D) are upstream from A (Fig.3(b)). Consequently, node A sends each of them a control packet. Node D replies that it has an exit node E. Node A therefore records the RAP and source-routes the packet along the route A-D-E.

4. DETAILED DESCRIPTION OF THE SPF-EE ALGORITHM

We now describe the route computation and packet forwarding procedures in some detail. A more formal ver-



(a)



(b)

Fig.3: Sink Trees for Node C and Node F

sion of the entire algorithms are presented in Appendix.

Route Computation

In the SPF-EE algorithm, each node maintains a routing table with one row per destination, as in Fig.4(a). The first column gives the next-hop along SP. The second column is dedicated to the APs. If an AP is not available, it is left blank. In addition to this, there is a RAP table for maintaining the the RAPs (Fig.4(b)).

The APs can be easily derived from the topological database within the node. They are calculated at the time of route updating along with the SPs. The calculation of RAP involves communications with other nodes. They are generated at an event-driven basis.

In theory, the maximum number of possible APs equals to the number of neighbors other than the one along the shortest-path. However, as emergency exits, they are much less frequently used than the SPs. Thus only one AP per destination are calculated at the time of route updating. Nevertheless, any AP or RAP can be calculated dynamically at the time of request.

When a node x receives a routing update, it first calculates the SPs for all the destinations by using the SPF algorithm detailed in [McQU80]. The result is a routing tree rooted at node x (denoted by $TREE_x$) and the first column of routing table for the next-hops of the SPs. It then derives the routing trees for each of its neighbors y $\in N_x$, where N_x is the set of neighbors of node x.

When the routing tree for node x is known, the calculation of the routing tree for an adjacent node y only requires an incremental calculation rather than a complete re-calculation [McQU80]. When the routing tree for node x is converted to the routing tree for node y, any nodes in the subtree rooted at node y will not be repositioned as those nodes are already at minimum delay. Therefore only nodes that are not on the subtree rooted at node y need to examine their neighbors to see if there is a shorter path.

For the routing tree $TREE_y$, y $\in N_x$, if destination node z is not on the subtree rooted at node x, and node y is not the next-hop of the SP for it, node y is the exit for destination z (see Fig.1(a)).

If no APs are found for destination z, a RAP will be established when the length of outgoing queue to destination z exceeds a threshold.

To establish a RAP to destination z, the initiating node x sends a *query* message (denoted by MSG(query,x,z)) to each of its neighbors r (r $\in N_x$, r $\neq NS_z$)(see Fig.1(b)). The message contains a message ID, addresses of node x and z. Node x then marks entry for destination z in the RAP table as *waiting*, which prevents further query messages being sent before it receives a reply. Upon receiving MSG(query,x,z), node r checks in the routing tree $TREE_m$ (m $\in N_r$, m $\neq$ x) whether destination z is on the subtrees looted at node x and r. If it is not, node m can be the exit for destination z.

Node r records the addresses of node x, z and m in the RAP table, includes the addresses of node r and m into a *reply* message (denoted by MSG(reply,x,z)) and sends it back to node x. Otherwise if no exits are found, node r includes its address and continues to propagates to its neighbors.

	SP	AP
A	-	-
B	B	D
C	B	D
D	D	B
E	F	D
F	F	-

(a)

Source	Destination	Intermediate Hops	Exit
A	F	D	E

(b)

Fig.4: Routing Table for Node A

If an exit is not found within a number of hops, the search is given up because a better exit is likely to be found along other paths or the exit is too far away or even nonexistent. If no RAPs can be found at all, the entry marked busy will effectively prevent further query messages being sent until a routing update is received and the RAP table is reset.

When node x receives a reply, it records the route to the exit in the RAP routing table. Node x may receive more than one reply messages for a query message, it only records the first one it receives and ignores the rest.

It should be clear now that the RAPs are maintained by both the initiating node x and the exit node y, and the RAP table records both the RAP entries and the exit entries. It any of the two nodes receives a routing update, the RAP table should be reset to prevent inconsistent RAPs and to allow new RAPs to be established. When node y receives a routing update, it checks each entry in the RAP table. If the source of one entry is different from the current node, ie. it is an exit record, it sends a *reset* message (denoted by MSG(reset,x,z)) to the source node and clears that entry. When node x receives a reset message from node y, it clears the entry in the RAP table that matchs node x, z, y and m.

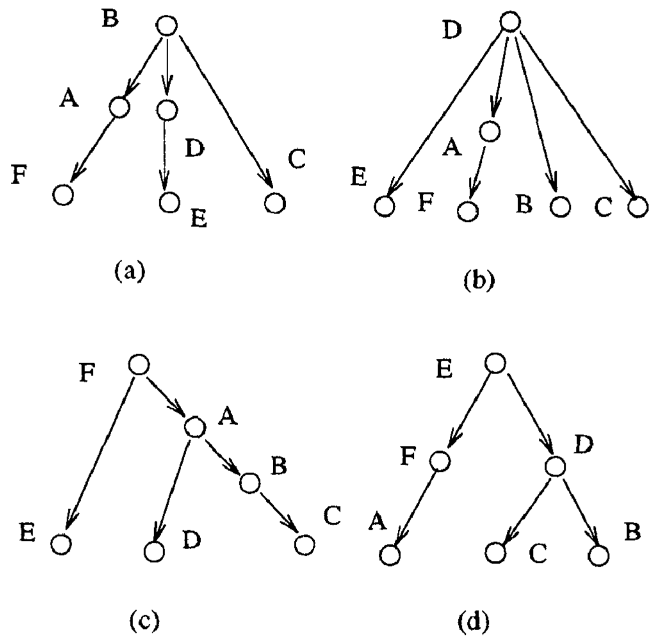


Fig.5: Routing Trees for Node B, D, E, F

Fig.5 shows the routing trees for node B, D, E and F. Take routing tree for node B for example, node D and E are not on the subtree rooted at node A, and node B is not their next-hop for the SP. Therefore node B is the next-hop of the APs for destination D and E. After all APs are filled into the routing table (Fig.4(a)), only destination F does not have an AP entry. When the link AF is congested, node A sends a query message to node D and B. After checking the routing tree for node E, node D finds an exit node E and replies back to node A. Node A records a RAP A-D-E for destination F.

Packet Forwarding

The packet forwarding in the SPF algorithm is straightforward: looking up the next-hop in the routing table and transmitting the packet to the output line. In the SPF-EE algorithm, however, the choice of the next-hop depends on the length of the outgoing queues. There are two predetermined parameters for traggering the use of APs or RAPs: T_s (threshold of queue length to node NS_z) and T_a (threshold of queue length to node NA_z).

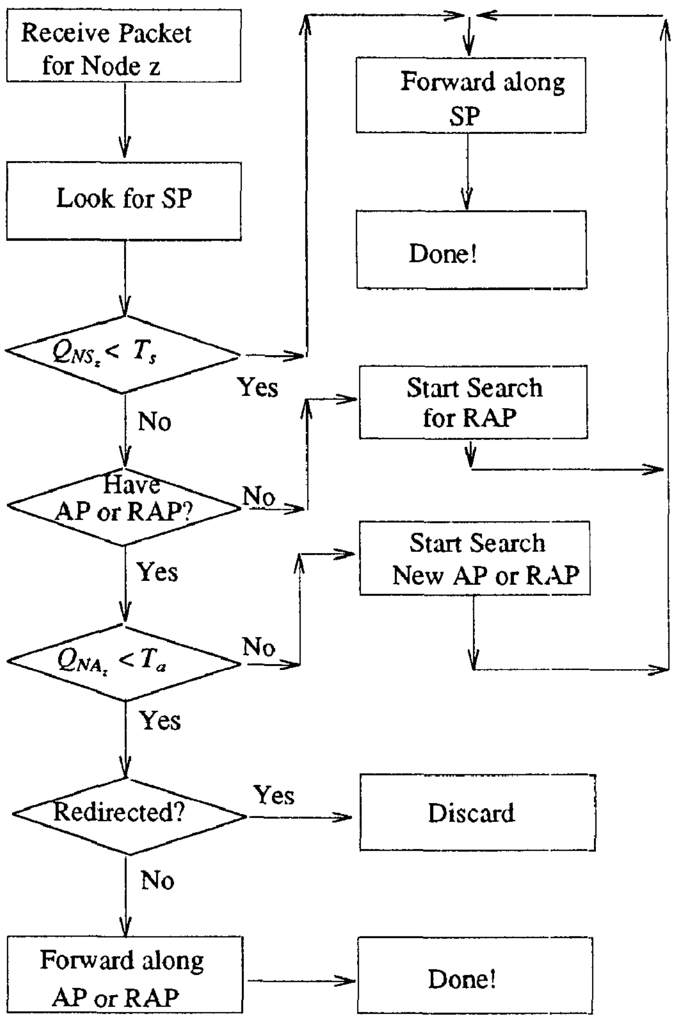


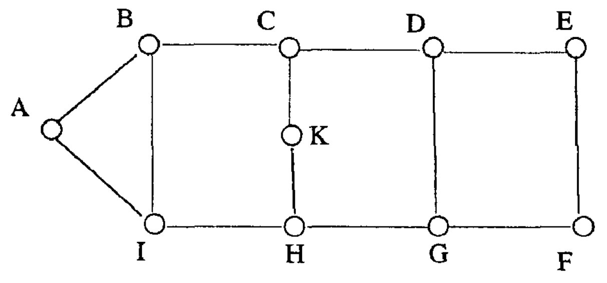
Fig.6: Forwarding Algorithm

Fig.6 shows a flowchart of the forwarding algorithm. When a node receives a packet, it looks up the next-hop along the SP and checks the outgoing queue length to this neighbor. If the queue length is smaller than T_s, the packet is transmitted to the next-hop. If the queue length exceeds the threshold, the node then checks if there are APs or RAPs available for the destination. If no APs or RAPs are found, the node initiates query messages to search for RAPs and forwards the packet along the SP. If there is one AP or RAP available, it checks the outgoing queue length of the AP or RAP. If the queue length exceeds T_a, the node starts search for a new AP or RAP and transmits the packet along the SP. Otherwise the node can transmit the packet along the AP or RAP. However, if the packet has already been transmitted along an AP or RAP by other nodes, a loop might be formed when the packet is sent back to where it comes from. This problem can be solved by marking explicitly the packet that is transmitted along an AP or RAP as a *redirected packet*. Before transmitting a packet along an AP or RAP, the node first checks the packet whether it is a *redirected packet*. If it is, the packet is discarded. Otherwise the packet is marked as *redirected packet* and transmitted along the AP or RAP. The fact that packets can be transmitted only once along an AP or RAP ensures that loops can not be formed by the redirected packets. Discarding the redirected packets in the face of congestion also reduces the effects of redirected traffic on other users.

5. PERFORMANCE COMPARISON

In this section, we present some simulation results on the performance of the SPF-EE algorithm. Since the SPF-EE algorithm is an attempt to eliminate certain problems with the SPF algorithm under heavy and dynamic traffic, our intention of the simulation is to compare the performance of the SPF-EE algorithm and the SPF algorithm, and to demonstrate the improvement of the SPF-EE algorithm over the SPF algorithm with regards to the problems discussed in section two.

Fig.7 shows the topology and some environment parameters used in the simulation. Experiments use both the ill-behaved TCP sources (TCP without congestion control) and the well-behaved TCP sources (TCP with slow-start, timeout-based congestion control and exponential retransmission backoff described in [JACO88].).



Link Speed: 200 KB/sec, Maximum Queue Length: 10
Propagation Delay: 50 us, T_a and T_s: 5

Fig.7: Simulation Topology and Environment Parameters

Maximum Flow

In the SPF algorithm, routing decisions are made at the time of route updating and remain unchanged throughout one route updating period (eg. 10s in SPF). During this period, any packets to a particular destination follows the same route. In the SPF-EE algorithm, the actual route that a packet is traveled is determined by individual nodes at the time of forwarding. A node may forward packets to both a SP and an AP (or RAP) in an alternate way. Therefore it can send packets at a speed greater than the capacity of one route.

Fig.8 shows the sending sequence number of a well-behaved TCP connection between node C and H with an excessive window size (15360KB).

With the SPF algorithm, the SP between node C and H oscillates among CBIH, CJH and CDGH and the throughput saturates around 20 KB/sec. With the SPF-EE algorithm, however, after some initial stage, all the three paths (CBIH, CJH and CDGH) are open for this connection and the throughput approaches 60 KB/sec, which is the theoretic limit between the two nodes. The gaps in the curve indicate packet drops and retransmits.

Due to the effect of the congestion control in the TCP sources, the paths between node C and H are not severely congested. Fig.9 shows the sending sequence number of a ill-behaved TCP connection. The SPF performs much worse without congestion control in the TCP source. Congestion and retransmission severely reduces the throughput. The curve for the SPF-EE in Fig.9 is almost the same as the one in Fig.8 except the sharp rise at the beginning in Fig.9, which is due to the fact the ill-behaved TCP source does not have the slow-start mechanism.

Oscillation

The SPF-EE algorithm can drastically reduce the negative effects of oscillation. With the SPF-EE algorithm, when the outgoing queue of a node exceeds the threshold, the node opens a new channel for the accumulated packets, which effectively reduces the congestion and queueing delay. Thus, the reported link delay does not oscillate so dramatically.

When links are congested, the queueing delay increases significantly. At next route updating, it is likely that the reported delay of the links will be so high that those links will be excluded from all routing trees of any sources. As a result, those links will be abandoned for one entire route updating period while resources are in shortage. Moreover, those links will report low delays when next route updating is due and become conested again. With the SPF-EE algorithm, those links that become unattractive to all sources will still be used as APs. As in Fig.8, although the SP between node C and H still oscillates between CBIH, CJH and CDGH, three paths are all in use.

Overhead and Responsiveness

The SPF-EE algorithm needs more storage and CPU resources in the node than the SPF algorithm. The routing table in the SPF-EE is larger than the one in SPF as there one column for the APs and additional table for RAPs. But the increase is around 50%. For each route updating, the SPF-EE calculates routing trees for itself and its neighbors, thus it needs approximately D times computation as the SPF, where D is the average degree of the nodes. The messages for establishing the RAPs are rather small and the information exchanges are confined in local areas. Once a RAP is established, it is cached until next route updating point. Therefore, the additional overhead is negligible as compared with the broadcasting of route updating.

On the other hand, the SPF-EE algorithm can reduce the frequency of route updating and therefore significantly decrease the computation and communication overhead. With dynamically changing traffic, it is not possible to obtain an accurate estimate of average delay within very

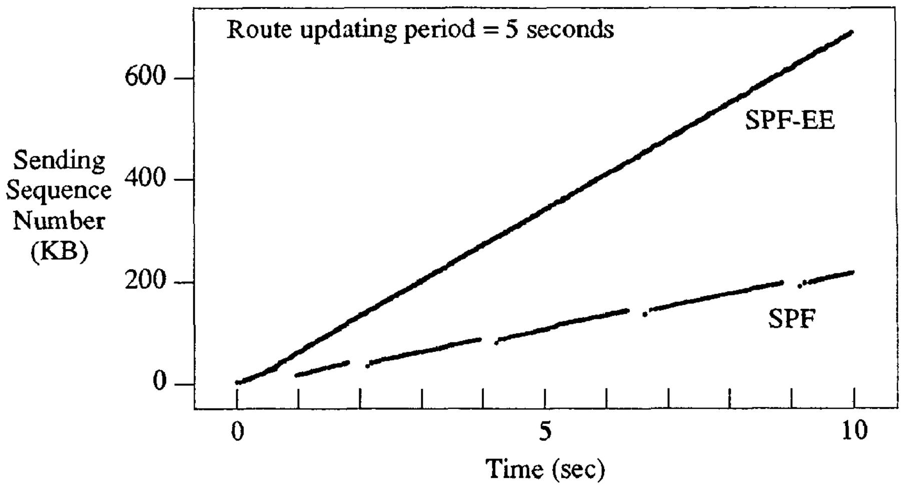


Fig.8: Sending Sequence Number of a well-behaved TCP Connection

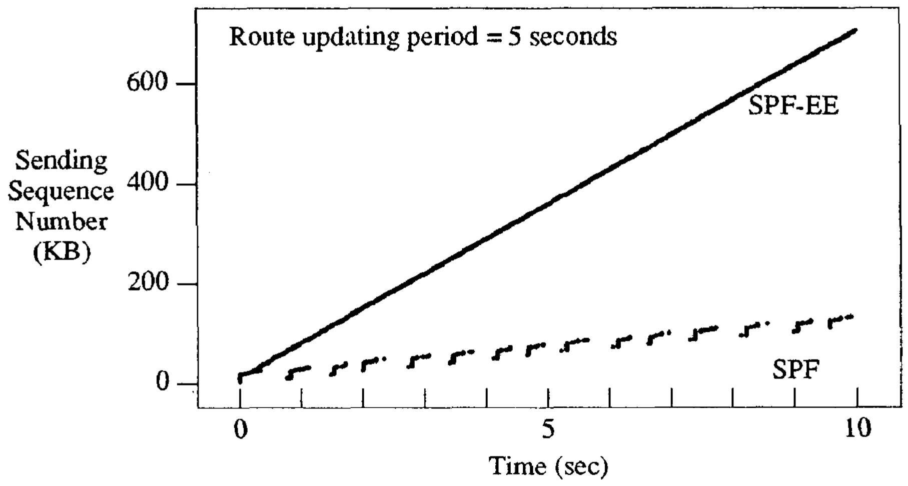


Fig.9: Sending Sequence Number of an ill-behaved TCP Connection

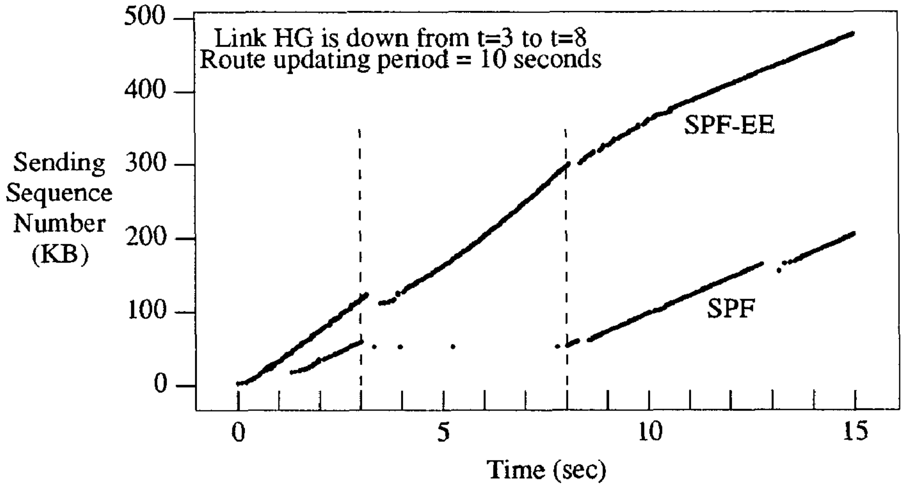


Fig.10: Fault Tolerance

short time. With the SPF-EE algorithm, the route updating period can be increased as the momentary fluctuation of the traffic is handled by APs and RAPs. The routing table will be updated only when long-lasting topological or traffic changes have taken place. Many superficial routing updates in the SPF algorithm traggered by traffic fluctuation are eliminated.

With the SPF-EE algorithm, the increase in the route updating period will in no way reduce the responsiveness. In fact, with the SPF-EE algorithm the nodes respond to the traffic on a packet-by-packet basis and are able to find alternate paths when necessary. The dilemma of overhead and responsiveness is solved by updating the routing table to adapt the topological changes and providing alternate paths to cope with the fluctuation of traffic.

Congestion Control and Fault Tolerance

Congestion occurs when the traffic load and the resources available mismatch at some points in the network. Most of the congestion control mechanisms currently deployed in computer networks are to reduce the traffic at the source or to prevent traffic from entering the network. However, traffic distribution over the network is often uneven. Congestion can also be eased by dispersing traffic to less congested areas without reducing traffic rate at the source. In the SPF algorithm, any routing changes require global route updating, which is often too slow to handle dynamic traffic fluctuations. The SPF-EE algorithm provides mechanisms to react to traffic conditions at a packet-by-packet basis. In the face of link or node failures, with the SPF algorithm the traffic has to wait until the failures are detected and route updating is performed before it can be routed to a new path. With the SPF-EE algorithm, traffic can be routed immediately along the APs until next route updating is due. Therefore, the SPF-EE algorithm provides greater transparency to the network users.

Fig.10 shows the result of an experiment with link HG is down for 5 seconds. A well-behaved TCP connection is established between node A and node F in Fig.7. About 3 seconds after route updating when the routing table indicates that path AIHGF is used, link HG goes down and it comes up again another 5 seconds later. Fig.10 shows that with the SPF the traffic flow is completely prevented during the downtime. While with the SPF-EE, the connection recovers very quickly and the traffic is routed along the alternate path HJCDGF. If the packets are accumulating along the AIHGF, node A may simply forward via node B.

6. DISCUSSIONS

In this paper, we have examined the problems of the SPF algorithm in a dynamic environment and concluded that route updating is often too slow and too costly to deal with traffic fluctuations and it may lead to instability under heavy traffic load. We then presented a new routing algorithm SPF-EE which can provide alternate paths for the congested traffic without route re-computation.

The SPF-EE algorithm attempts to redirect congested traffic to where resources are available therefore it is most effective when traffic distribution is uneven. The SPF-EE algorithm should be used along with other congestion control algorithms (eg. [JACO88]) which are able to reduce the traffic at the source when the network is experiencing congestion, so that the network wide congestion will not occur and the alternate paths are only used as emergency exits.

When the traffic is dispersed to alternate paths, it will affect traffic from other sources. The SPF-EE algorithm enables traffic sources use more resources than that is allowed in the SPF algorithm. However, more sophisticated queueing or dropping algorithms at the nodes can minimize those effects. For example, the redirected packets may have a lower priority in forwarding or be first dropped when the buffer is overflowed.

The simulation we have done is limited in scope. Experiments on more complex topology and traffic patterns are needed to further study the behavior of the SPF-EE algorithm and their implications on network performance.

ACKNOWLEDGEMENT

The authors would like to thank the anonymous referees for their valuable comments.

REFERENCES

[McQU80] J. McQuillan, I. Richer, E. Rosen, *The New Routing Algorithm for the ARPANET*, IEEE trans. on comm., Vol. COM-28, No. 5, pp 711-719, May 1980.

[ROSE80] E. Rosen, *The Updating Protocol of ARPANET's New Routing Algorithm*, Computer Networks, Vol. 4, pp 11-19, 1980.

[BERT82] D. Bertsekas, *Dynamic Behavior of Shortest Path Routing Algorithms for Communication Networks*, IEEE trans. on automatic control, Vol. AC-27, No.1, Feb. 1982.

[WANG90] Z. Wang, J. Crowcroft, *Analysis of Shortest-Path Routing Algorithms in a Dynamic Environment*, Research Notes RN/90/31, Computer Science Dept, University College London, March 1990.

[SEEG86] J. Seeger, A. Khanna, *Reducing Routing Overhead in a Growing DDN*, in Proc. of MILCOM'86, pp 15.3.1-15.3.13, Oct. 1986.

[KHAN89] A. Khanna, J. Zinky, *The Revised ARPANET Routing Metric*, in Proc. fo ACM SIGcomm'89, pp 45-56, Sept. 1989.

[JACO88] V. Jacobson, *Congestion Avoidance and Control*, in Proc. of ACM SIGcomm'88, pp 314-329, 1988.

APPENDIX

This Appendix give a complete description of the SPF-EE algorithm for route computation and packet forwarding. In this description, node x is the node where the algorithm is running. LIST is a list structure for the computation. D^{y}_{xz} denotes the total length of the path from node x to node z via node y. D_{xz} denotes the length of the shortest path from node x to node z that is known at the time of computation. L_{xy} denotes the length of the link between node x and y. T_x and T'_x represent respectively the routing table and RAP table for node x.

Procedure 1 (route computation for SP)

```
when node x receives a update and $L_{im}$ is changed by $\Delta L_{im}$
begin
  if $TREE_x$ does not exist  then
     place $D^{x}_{xx}$ on LIST;
  else
     begin
       if link im is in $TREE_x$ then
          set $\Delta = \Delta L_{im}$;
       else
          begin
            set $\Delta = D_{xi} + L_{im} + \Delta L_{im}$ (mi $D_{xm}$;
            if $\Delta \geq 0$ then
               stop;
          end
     end
  for each node j in the subtree $TREE_m$ do
     set $D_{xj} = D_{xj} + \Delta$;
  for each node j in the subtree $TREE_m$ do
     begin
       if $\Delta > 0$ then
         begin
           get $D^{q}_{xj}$ so that $q \in N_j$, $q \notin TREE_m$, $D^{q}_{xj} = D_{min}$, $D^{q}_{xj} < D_{xj}$;
           if $D^{q}_{xj} \neq \emptyset$ then
              place $D^{q}_{xj}$ on the LIST;
         end
       else
         begin
           get $D^{j}_{xq}$ so that $q \in N_j$, $q \notin TREE_m$, $D^{j}_{xq} = D_{min}$, $D^{j}_{xq} < D_{xq}$;
           if $D^{j}_{xq} \neq \emptyset$ then
              place  $D^{j}_{xq}$ on the LIST;
         end
  while LIST is nonempty do
     begin
       get $D^{l}_{xn}$ so that $D^{l}_{xn} \in$ LIST, $D^{l}_{xn} = D_{\min}$;
       remove $D^{l}_{xn}$ from LIST;
       place node n on $TREE_x$ so that node l is the predecessor;
       for each node k, $k \in N_n$ do
         begin
           if k is in $TREE_x$ and $D_{xk} > D^{n}_{xk}$ then
              begin
                remove node k from $TREE_x$;
                place $D^{n}_{xk}$ on LIST;
              end
           else if $D^{v}_{xk}$ ($v \neq n$) is in LIST and $D_{xk} > D^{n}_{xk}$ then
              begin
                remove $D^{v}_{xk}$ from LIST;
                place $D^{n}_{xk}$ in LIST;
              end
           else if $D^{v}_{xk}$ (v ≠ n) is not in LIST then
              place $D^{n}_{xk}$ in LIST;
        end
     end
        update $T_x$;
end
```

Procedure 2 (route computation for AP)
when node x receives a update and L_{im} is changed by ΔL_{im}

```
begin
  for each entry in $T^{r}_{x}$ do
     begin
       if source is not node x then
          send MSG(reset, x, z) to source node;
     end
  for each node y, $y \in N_x$ do
     begin
       place node y at the root of $TREE_y$;
       for each node j in the subtree $TREE_x$ do
         begin
           get $D^{q}_{yj}$ so that $q \in N_j$, $q \notin TREE_x$, $D^{q}_{yj} = D_{min.}$ $D^{q}_{yj} < D_{yj}$;
           if $D^{q}_{yj} \neq \varnothing$ then
              place $D^{q}_{yj}$ on the LIST;
         end
    while LIST is nonempty do
       begin
         get $D^{l}_{yn}$ so that $D^{l}_{yn} \in$ LIST, $D^{l}_{yn} = D_{\min}$;
         remove $D^{l}_{yn}$ from LIST;
         place node n on $TREE_y$ so that node l is the predecessor;
         for each node k, $k \in N_n$ do
           begin
             if k is in $TREE_y$ and $D_{yk} > D^{n}_{yk}$ then
                begin
                  remove node k from $TREE_y$;
                  place $D^{n}_{yk}$ on LIST;
                end
             else if $D^{v}_{yk}$ (v ≠ n) is in LIST and $D_{yk} > D^{n}_{yk}$ then
               begin
                 remove $D^{v}_{yk}$ from LIST;
                 place $D^{n}_{yk}$ in LIST;
               end
             else if $D^{v}_{yk}$ (v ≠ n) is not in LIST then
                place $D^{n}_{yk}$ in LIST;
           end
       end
     end
       update $T_x$;
end
```

Procedure 3 (packet forwarding)
when node x receives a packet for node z

```
begin
  get $NS_z$ from $T_x$;
  if $Q_{NS_z} < T_s$ then
```

```
      send along SP;
  else
    begin
      get $NA_z$ from $T_x$ or $T^r_x$;
      if $NA_z = \emptyset$ and RAP entry for x, z not marked busy then
        begin
          send MSG(query, x, z) to $N_x$;
          mark RAP entry for x, z busy;
          send packet along SP;
        end
      else if $Q_{NA_z} < T_a$ then
        if packet is marked redirected then
          discard the packet;
        else
          begin
            mark the packet as redirected;
            send packet along AP or RAP;
          else
      else
        begin
          send MSG(query, x, z) to $N_x$;
          send packet along SP;
        end
    end
end
```

```
Procedure 4 (search for RAP)
when node w receives MSG(query, x, z)
begin
  for each $TREE_n$, n $\in N_w$ do
    begin
      if destination z is not on $TREE_w$ and $TREE_x$ then
          begin
            record x, z, w and n in $T^r_w$;
            send MSG(reply, x, z) to node x;
          end
      else
          send MSG(query, x, z) to $N_n$;
    end
end
```

```
Procedure 5 (search for RAP)
when node x receives MSG(reply, x, z)
begin
  if an entry for node x and z does not exist in $T^r_x$ then
    record RAP path in $T^r_x$;
end
```

```
Procedure 6 (search for RAP)
when node x receives MSG(reset, x, z)
begin
  if an entry for the path exists in $T^r_x$ then
    clear this entry;
end
```