Hybrid Flooding and Tree-based Broadcasting for Reliable and Efficient Link-state Dissemination

Turgay Korkmaz
Department of Computer Science
The University of Texas at San Antonio
San Antonio, TX 78249
korkmaz@cs.utsa.edu

Marwan Krunz
Department of Electrical & Computer Engineering
University of Arizona
Tucson, AZ 85721
krunz@ece.arizona.edu

Abstract— **Current link-state routing protocols (e.g., OSPF) use flooding to disseminate link-state information throughout the network. Despite its simplicity and reliability, flooding incurs unnecessary communications overhead since nodes may receive multiple copies of the same advertisement. This extra overhead becomes an issue in the context of quality-of-service (QoS) routing, where link state is dynamic and needs to be advertised frequently. The advertisement overhead can be significantly reduced by using tree-based broadcasting approaches. Although several of these approaches have been proposed in the literature, they are not used in practice because of their complexity and/or unreliability. We propose a new link-state dissemination approach that combines the best features of flooding and tree-based broadcasting. Our hybrid approach is particularly suited for "dynamic" link metrics (e.g., available bandwidth). It uses periodic flooding to advertise topology changes and first-time LSAs (link-state advertisements), and uses tree-based broadcasting to disseminate subsequent refresh LSAs. The broadcast trees in our approach are constructed dynamically during the flooding of the first LSA, without the need for the complex algorithms of previous tree-based approaches. Two versions of our dissemination approach are presented, with one being more suitable for networks with frequent topological changes. We prove the correctness of our approach and contrast its communications overhead with flooding and pure tree-based broadcasting. The results indicate that our hybrid approach has a significantly lower overhead than flooding; yet it enjoys the simplicity, reliability, and fast convergence of flooding. Finally, we outline how OSPF can be extended to support the proposed dissemination approach.**

I. Introduction

In link-state routing, every node in a routing domain tries to maintain an accurate "map" of the underlying network. It does that by encoding the state information related to its outgoing links into a link-state advertisement (LSA) packet and disseminating this LSA throughout the network. For simple and reliable dissemination, existing link-state routing protocols (e.g., OSPF [1], [2] and PNNI [3]) use flooding, in which an incoming LSA is forwarded to all neighbors except the one from which the LSA is received. Despite its simplicity and reliability, flooding involves unnecessary communications, causing inefficient use of resources. To minimize the communications overhead, current routing protocols use small LSAs and large update (refresh) intervals. For instance, in OSPF a single, relatively static, "cost" metric is advertised periodically every 30 minutes. Such infrequent dissemination is sufficient for a best-effort service. To support emerging QoS-oriented services (e.g., DiffServ), routing protocols will need to be extended to disseminate additional link state information, including available bandwidth, delay, and jitter. With more parameters being advertised, the size of the LSA inevitably gets larger. Moreover, some of these parameters (e.g., available bandwidth) are quite dynamic, and thus must be frequently disseminated (e.g., using triggered updates [4]) to provide an accurate representation of the underlying network. In these scenarios, the overhead of flooding can be quite excessive.

To achieve efficient state dissemination, researchers have investigated tree-based broadcasting approaches [5], [6], in which LSAs are forwarded over broadcast trees such that every node receives exactly one copy of each LSA. While tree-based broadcasting reduces the dissemination overhead, it introduces a challenging problem, namely how to determine and maintain *consistent* broadcast trees throughout the network. Previously proposed solutions for this problem rely on complex algorithms and protocols (see Section II for details), making them impractical for real networks.

In this paper, we propose a hybrid mechanism that combines the best features of flooding and tree-based broadcasting. The proposed mechanism alternates between flooding and tree-based broadcasting modes. In the flooding mode, "first-time" LSAs are flooded to establish the broadcast trees, over which subsequent (refresh) LSAs are advertised in the tree-based broadcasting mode. The hybrid mechanism is shown to achieve a significant reduction in the communications overhead compared to flooding; yet, it maintains the simplicity and reliability of flooding. Compared to pure tree-based broadcasting, the proposed mechanism incurs a slight extra overhead, but this overhead is overshadowed by the simplicity of this mechanism and its amenability to practical application in real networks. Furthermore, in contrast to previous tree-based broadcasting mechanisms, in which the broadcast trees are determined based on the hop count, ours uses the minimum delay experienced during the flooding of a first-time LSA to compute the broadcast trees. As a result, it enjoys the same fast convergence of flooding.

In principle, the proposed hybrid mechanism can be used with any link-state routing protocol. However, for the sake of concreteness, we investigate it in the context of the OSPF protocol and its QoS extensions [7], [8]. We explain how OSPF can be extended to support the hybrid scheme. This entails: (1) redefining two of the currently unused bits in the LSA header, (2) adding a table in each router for maintaining information relevant to the broadcast trees, and (3) adding and/or modifying some of the steps in the "flooding procedure" of OSPF.

The rest of this paper is organized as follows. In Section II, we give background information on flooding in OSPF and review the literature on tree-based broadcasting. In Section III we present the basic version of the proposed hybrid mechanism. In Section IV we present a modified version of this mechanism, which is suitable for highly dynamic topologies. We also describe how to extend OSPF to support the proposed hybrid mechanism. In Section V we analyze the communications overhead of the hybrid mechanism and contrast it with flooding and pure tree-based broadcasting. Finally, in Section VI we conclude the paper and give some directions for future research.

II. Background and Related Work

Flooding in OSPF

Flooding is used in OSPF to disseminate the link-state information to all routers in the same domain. Each router periodically generates LSAs representing the parameters of its outgoing links and sends these LSAs to all of its neighbors. Receiving routers then forward the LSAs to their neighbors except the ones from which the LSAs have been received. For reliability purposes, each LSA is acknowledged. This flooding process allows every node to acquire the same map of the network.

This work was supported by the National Science Foundation under grants ANI 9733143, CCR 9979310, and ANI 0095626.

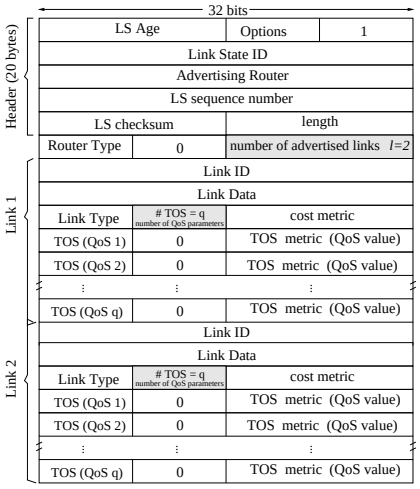

Fig. 1. Structure of an OSPF router-LSA with two advertised links.

OSPF uses five types of LSAs. For a network with point-to-point links, only one of these types, known as *router-LSA*, is used. The basic structure of a router-LSA is shown in Figure 1. Every router-LSA starts with a 20-byte header that contains specific information to uniquely identify the LSA. The rest of the LSA contains the values of the "cost" metric and any additional QoS parameters that are associated with the outgoing links of the LSA's originator. Note that OSPF currently uses a single metric, but the type-of-service (TOS) field in OSPF, which has not been much used in the past, can be redefined to advertise multiple link parameters (see [7] for details). Consider the shaded fields in Figure 1. In this example, the number of advertised links is indicated by l. The link-state information is then repeated l times with different values. Each link has some identification information followed by the link parameters. The number of link parameters is indicated by q. Each parameter is encoded using a four-byte field that includes the parameter name and its value. Hence, the size (in bytes) of a router-LSA originating from node u is given by $S(u) \stackrel{\text{def}}{=} 20+4+l(u)(12+4q)$, where $l(u)$ is the number of advertised outgoing links from node u (i.e., $l(u) = \deg(u)$). Note that an LSA acknowledgment (ACK) consists of only a 20-byte header.

Tree-based Broadcasting

State dissemination based on tree-based broadcasting appears in the literature in two forms: a single broadcast tree (SBT) and multiple broadcast trees (MBT). In the SBT approach, all nodes compute a common broadcast tree (e.g., a spanning-tree), and every node marks its own links on that tree. Every node then receives LSAs via one of its marked links and forwards the LSAs through its other marked links. The SBT approach has two main disadvantages. First, it results in an unbalanced load distribution since LSAs are sent over a fixed subset of the network links (i.e., the links that belong to the broadcast tree). Second, it is quite possible for nodes that are neighbors according to the network graph to lie far away from each other on the broadcast tree, a situation that delays the convergence of the routing protocol. In [9] the authors explored the viability of the SBT approach for state dissemination in the PNNI protocol. They provided a distributed spanning-tree algorithm for determining the broadcast tree. However, finding this tree and maintaining it in a consistent manner involves complex operations such as exchanging extra control packets besides LSAs and executing the spanning-tree algorithm in a distributed manner.

In the MBT approach, every node has its own broadcast tree (e.g., a shortest path tree). For illustration, consider the trees originating from nodes 1 and 3 in Figure 2. The LSAs originating from a given node are disseminated over that node's broadcast tree. For example, node 1 generates an LSA and sends it to nodes 2 and 3. However, only node 2 forwards this LSA to node 4 (according to the broadcast tree of node 1). To disseminate LSAs over their originators' broadcast trees, every node needs to know the broadcast trees of all other nodes. This can be done

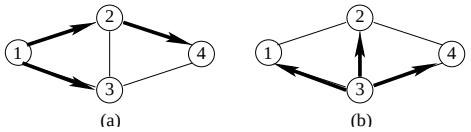

Fig. 2. Multiple broadcast trees originating from nodes 1 and 2.

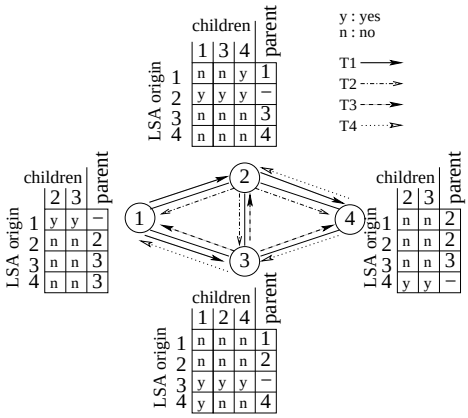

Fig. 3. Representing the parent-children relationships of the broadcast trees in the MBT approach.

as follows [5]. Every node determines its parent and children on every broadcast tree and stores these parent-children relationships in a table. Figure 3 illustrates an example. Let T_i be the broadcast tree originating from node i, $i = 1, 2, \ldots, n$, where n is the number of nodes in the network. Consider the representation of T_1 at node 2. Since node 2 has only one child on T_1 (namely, node 4), it marks node 4 as its child in the first row of its table. So whenever node 2 receives an LSA originating from node 1, it forwards this LSA to node 4 only. Node 2 also stores its parent on T_1 (namely, node 1) in the first row of its table. Since LSAs are disseminated over different shortest-path trees, the MBT approach provides some form of load balancing. The convergence time of this approach is also faster than that of the SBT approach.

A key issue in the MBT approach is how to determine the broadcast trees in a distributed manner. Previously proposed approaches achieve that through additional control packets (besides the LSAs) and by relying on protocols that execute some variant of the shortest path algorithm [5], [6]. This complicates the establishment and maintenance of consistent broadcast trees. The main objective of these complicated mechanisms is to always disseminate LSAs over the broadcast trees. In [5] the authors addressed the issue of determining broadcast trees while the topology information is still being disseminated over these trees. In [6] the authors considered the idea behind reverse-path forwarding (RPF) in [10] and proposed a new topology dissemination protocol called TBRPF, in which broadcast trees are computed based on full topological information received over the broadcast trees themselves. In TBRPF, every node executes Dijkstra's algorithm to determine a reverse minimum-hop tree, and then exchanges some information with neighbors to determine its parent and children from the standpoints of other nodes. Although TBRPF provides more reliability than other existing methods, it suffers from the overhead associated with

computing the trees and communicating with neighbors whenever a topological change occurs.

III. Hybrid Dissemination Mechanism

In this section, we present our Hybrid Flooding and Tree-based Broadcasting (HFTB) mechanism. Recall that in OSPF [2], LSAs containing the values of the cost metric are periodically flooded every 30 minutes. While this duration is sufficient for the relatively static cost metric, other link parameters may change several times within the 30-minute period. These parameters need to be frequently disseminated, e.g., using triggered updates. The objective of HFTB is to disseminate triggered LSAs using tree-based broadcasting while continue to use flooding for disseminating the cost metric and connectivity information.

HFTB is similar to previous MBT approaches in the sense that every node maintains the same parent-children relationships, as shown in Figure 3. However, in contrast to previous MBT approaches, HFTB uses flooding of the *first* LSA in each 30-minute update interval to establish the broadcast trees, which are then used to disseminate subsequent "refresh" LSAs generated *within* the update interval. No extra control packets or complex protocols are needed to establish the broadcast trees.

The broadcast trees are established as follows. Let LSA_u denote a flooded LSA that was generated by some node u. Suppose that LSA_u arrives at some node i for the first time through node j, as shown in Figure 4. Node i selects node j as its parent from

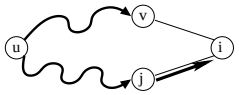

Fig. 4. Establishing broadcast trees in the HFTB approach.

the standpoint of node u and acknowledges node j. When node j receives the acknowledgment, it records node i as its child from the standpoint of node u. If LSA_u arrives at node i again via another node v, then node i acknowledges the LSA as in OSPF without establishing a new parent-child relationship. After the flooding of the first LSA_u, every node can determine its parent and children on the broadcast tree of node u (T_u). In contrast to previous approaches that establish the broadcast trees w.r.t. the minimum hop count, HFTB dynamically determines the broadcast trees w.r.t. the actual minimum delay, and broadcasts LSAs over these trees, suggesting that HFTB converges as fast as flooding.

*Theorem 1: (***Correctness of HFTB***) Consider a network $G = (V, E)$ with bidirectional links. Suppose that an arbitrary node $s \in V$ generates an LSA and floods it throughout the network. Furthermore, suppose that the LSA experiences some delay $d(u, v)$ when processed and forwarded from node u to node v. Using HFTB along with the flooding of the first LSA_s, T_s is established throughout the network in finite time (i.e., every node determines its parent and children on T_s). Moreover, T_s converges to a shortest-paths tree.*

Proof: Initially, T_s consists of only node s. So, the theorem is trivially true. Consider T_s after flooding LSA_s through some nodes in the network. In flooding, an arbitrary node u may receive the same LSA_s several times. However, node u forwards the incoming LSA_s to its neighbors only once, upon the first arrival of this LSA. In addition, node u selects its parent on T_s by acknowledging the node from which LSA_s was received for the first time. Subsequent arrivals of LSA_s at node u are acknowledged without establishing a parent-child relationship. Let $t[s]$ denote the time at which node s generates LSA_s, and $t[u]$ the time at which node u receives LSA_s for the first time along a path $p = \langle v_0 = s, v_1, \ldots, v_k = u \rangle$. Without loss of generality, we assume that $t[s] = 0$. So, $t[u] = \sum_{i=1}^{k} d(v_{i-1}, v_i)$.

To prove that T_s is a shortest-paths tree, we need to show that the path p does not contain any cycle and that p is the shortest path from s to u. The proof follows similar arguments to those used in [11, pages 523-525] for shortest-paths trees. However, instead of the process of relaxing a link (u, v) in [11], we consider the process of forwarding first-time LSA_s from node u to node v. So it is sufficient to show that forwarding LSA_s from node u to node v upon its first arrival is the same as relaxing the link (u, v) in the computation of a shortest-paths tree.

In computing the shortest-paths tree, a node u with minimum $t[u]$ is selected and every link (u, v) is considered for relaxation in a sequential manner. If $t[u] + d(u, v) < t[v]$, then link (u, v) is relaxed, i.e., the parent of node v is set to node u and $t[v]$ is set to $t[u] + d(u, v)$. In HFTB, paths are explored in parallel. So a node u starts forwarding LSA_s through (u, v) as soon as it receives LSA_s for the first time. In other words, a node u with minimum $t[u]$ is automatically selected in parallel and every link (u, v) is considered. If node v receives LSA_s for the first time via node u, then the parent of node v is set to node u, and $t[v]$ is set to $t[u] + d(u, v)$; this is the same as relaxing (u, v) in [11]. Otherwise, no parent-children relationship is established again. The rest of the proof follows the same proofs in [11, pages 523-525].

Finally, forming the broadcast tree T_s takes finite time since in flooding, every node receives LSA_s in a finite amount of time. ■

Figure 5 explains how HFTB can be integrated into the state dissemination procedure of OSPF. The first required modification to OSPF is to designate one of the unused bits of the Options field in the LSA header as *Flooding or Tree-based broadcasting* (FT)-bit. If this bit is set to 0, the LSA will be flooded throughout the network as in the standard OSPF. The second modification is to maintain a table at each node to record the parent-children relationships. The third modification is to add new steps into the flooding procedure of OSPF to establish the broadcast trees during the flooding of the first LSA in the 30-minute update interval. The subsequent LSAs generated within the 30-minute interval are broadcasted by setting the FT-bit to 1. The FT-bit is also used in LSA ACKs. In this case, if the FT-bit is 1, then this is an indication that the receiver has selected the sender as its parent; otherwise, the receiver has a different parent. Accordingly, the sender records the receiver as its child

```
Basic HFTB executed at node i
 Upon becoming operational
 1.  node i initializes its parent-children table
 2.  node i synchronizes its link-state database as in OSPF
 3.  node i goes into flooding mode
 Repeat every 30 minutes
 1.  node i goes into flooding mode
 2.  node i generates a new LSA that describes the state of its outgoing links
 Upon generating a new LSA
 1.  if node i is in flooding mode then
 1.1     set FT-bit in Options to 0
 1.2     send the LSA to all neighbors
 1.3     node i goes into tree-based broadcasting mode
 2.  else if node i is in tree-based broadcasting mode then
 2.1     set FT-bit in Options to 1
 2.2     send the LSA to all children of node i
 3.  end if
 Upon receiving an LSA originating from node u via node j
 1.  if FT-bit in Options is 0 then
 1.1     if the LSA is the most recent and
                 this is the first time it arrives at node i then
 1.1.1       send the LSA to all neighbors except node j
 1.1.2       table[LSA's originator].parent = node j
 1.1.3       send LSA ACK to node j by setting FT-bit in Options to 1
 1.2     else
 1.2.1       send LSA ACK to node j by setting FT-bit in Options to 0
 1.3     end if
 2.  else if FT-bit in Options is 1 then
 2.1     send LSA ACK to node j (FT-bit is set to 1)
 2.2     send the LSA to all children according to LSA's originator
 3.  end if
 Upon receiving an LSA ACK from node j
 1.  if FT-bit in Options is 1 then
 1.1     table[LSA's originator].children[node j] = yes
 2.  else if FT-bit in Options is 0 then
 2.1     table[LSA's originator].children[node j] = no
 3.  end if
 Upon failure of link (i, j)
 1.  for each originator node u do
 1.1     if table[originator u].parent is node j then
 1.1.1       table[originator u].parent = no parent
 1.2     end if
 2.  end for
 3.  send the LSA that indicates the link failure to all neighbors (FT-bit=0)
end HFTB
```

Fig. 5. Integrating HFTB into the OSPF protocol.

when it receives an ACK with an FT-bit of one.

IV. Enhanced Hybrid Mechanism

The previously discussed HFTB mechanism computes the broadcast trees once every 30 minutes. However, due to topological changes, particularly link failures, some broadcast trees may become disconnected shortly after they have been updated. If no action is taken to repair these trees, some nodes may not receive the up-to-date values of link parameters for at most 30 minutes. So taking no action (i.e., using the basic HFTB) could be a viable solution if the probability of a link failure is low and if the underlying path selection algorithms are capable of dealing with inaccurate state information. To maintain highly accurate routing information at every node, disconnected trees need to be repaired *within* each period. This is done using a slightly modified version of HFTB, which we refer to as *safe HFTB* (S-HFTB).

The idea behind S-HFTB is to use the basic HFTB while dynamically repairing disconnected broadcast trees. Note that when a link (j, i) goes down, the broadcast trees according to which node j is the parent/child of node i become disconnected. When node i receives a link failure message from the physical layer, it initializes its parent-child relationship with node j, disconnecting some trees at node i. If all nodes execute S-HFTB, then a disconnected tree at node i can be detected from another node v, as follows. Node v forwards only the header of an incoming LSA to neighbors that are not on the broadcast tree of the LSA's originator. When this header is received at some node i, this node checks whether it has a parent on the broadcast tree of the incoming LSA's originator. If no such parent exists, then node i is "disconnected" from the standpoint of the LSA's originator. So, node i selects node v, from which the header of the LSA was received, as its parent and asks this parent for the complete LSA. As an example, consider the broadcast tree originating from node u in Figure 4. Suppose that link (j, i) has gone down, disconnecting T_u. Furthermore, suppose that node v has just received LSA_u. Since node i is not a child of node v on T_u, node v sends the header of LSA_u to node i. Upon receiving this header, node i (which lost its parent on T_u due to the failure of link (j, i)) selects node v as its parent and sends an ACK message to node v, indicating the establishment of a new parent-child relationship. When node v receives this ACK, it records node i as a child and sends it the complete LSA_u. As

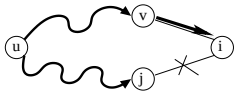

Fig. 6. Repairing broadcast trees in S-HFTB.

a result, the broadcast tree originating from node u is repaired, as shown in Figure 6.

S-HFTB can be integrated into OSPF, as illustrated in the pseudo-code in Figure 7. In the flooding mode (i.e., when the FT-bit of the incoming LSA is 0), S-HFTB and HFTB are identical. In the tree-based broadcasting mode, nodes using S-HFTB perform two tasks. First, a node has to forward the incoming LSA to its children, as in HFTB. Second, the node forwards only the header of the incoming LSA to the neighbors that are not children on the broadcast tree of the LSA's originator. This latter task, which is not present in HFTB, is used to repair disconnected trees. To integrate this enhancement into the OSPF protocol, we designate one more unused bit in the Options field of the LSA header as the Repair Tree (RT) bit (this is an addition to the modifications made above for the basic HFTB). This bit is set to 1 if the LSA contains the header only. Suppose that a given node i receives LSA_u with FT-bit=1 and RT-bit=1 via some node v. Node i first checks whether is has a parent on T_u. If not, it selects node v as its parent and sends an ACK message with FT-bit=1 and RT-bit=1 to node v. Upon receiving this

```
S-HFTB executed at node i
 Upon becoming operational or Upon failure of link (i, j)
     perform the same tasks as in Basic HFTB
 Repeat every 30 minutes
     perform the same tasks as in Basic HFTB
 Upon generating a new LSA
 1.  if node i is in flooding mode then
 1.1     set FT and RT bits in Options to 0
 1.2     send LSA_i to all neighbors
 1.3     go into tree-based broadcasting mode
 2.  else (node i is in tree-based broadcasting mode)
 2.1     set FT-bit to 1 and RT-bit to 0
 2.2     send LSA_i to all children of node i according to T_i
 2.3     set FT and RT bits to 1
 2.4     send LSA header to all neighbors that are not
             children of node i according to T_i
 3.  end if
 Upon receiving an LSA originating from node u via node j
 1.  if FT-bit is 0 then perform the same tasks as in Basic HFTB
 2.  else if FT-bit is 1 then
 2.1     if RT-bit is 0 then
 2.1.1       send LSA ACK to node j with FT-bit=1 and RT-bit=0
 2.1.2       send the LSA_u to all children of node i according to T_u (FT=1, RT=0)
 2.1.3       set FT and RT bits to 1
 2.1.4       send the header of LSA_u to all neighbors that
             are not children of node i according to T_u
 2.2     else if RT-bit is 1 then
 2.2.1       if table[ LSA's originator u].parent = NIL and
                 LSA_u is the most recent and
                 it arrives at node i for the first time then
 2.2.1.1         send LSA ACK to node j with FT-bit=1 and RT-bit=1
 2.2.1.2         table[LSA's originator].parent = node j
 2.2.2       else
 2.2.2.1         send LSA ACK to node j with FT-bit=0 and RT-bit=0
 2.2.3       end if
 2.3     end if
 3.  end if
 Upon receiving an LSA ACK from node j
 1.  if FT-bit in Options is 1 then
 1.1     table[LSA's originator].children[node j] = yes
 1.2     if RT-bit in Options is 1 then
 1.2.1       send the LSA to node j with FT-bit=1 and RT-bit=0
 1.3     end if
 2.  else if FT-bit in Options is 0 then
 2.1     table[LSA's originator].children[node j] = no
 3.  end if
end S-HFTB
```

Fig. 7. Integrating S-HFTB into the OSPF protocol.

ACK, node v records node i as its child and sends the complete LSA with FT-bit=1 and RT-bit=0 to node i. If node i already has a parent, it sends an ACK message with FT-bit=0 and RT-bit=0 to node v. As a result, disconnected trees are dynamically repaired throughout the network.

V. Performance Evaluation

In this section, we analyze the communications overhead of the proposed dissemination approaches and compare them with flooding and pure tree-based broadcasting. Assume that we have a network with n nodes and m links. Each node u periodically (every 30 minutes) generates a router-LSA and floods it as in the standard OSPF. Within the 30-minute period, each node may be triggered to generate additional router-LSAs that advertise the most recent values of the link-state parameters. Assume that the average number of triggered advertisements within a 30-minute interval is λ. Recall that the size of an ACK packet is 20 bytes while the size of an LSA packet originating from node u is $S(u) = 20 + 4 + l(u)(12 + 4q)$ bytes. If no link failure occurs in a given 30-minute interval, then the total communications overhead (TCO) of various state dissemination mechanisms can be computed (in bytes) as follows. For pure flooding, LSAs and ACKs are sent over every link. So the TCO of flooding is $\text{TCO}_{flooding} \stackrel{\text{def}}{=} (\lambda + 1) \sum_{u=1}^{n} m[S(u) + 20]$

In pure tree-based broadcasting, LSAs are disseminated over trees, each consisting of $n-1$ links. Thus, the TCO of tree-based broadcasting is $\text{TCO}_{tree} \stackrel{\text{def}}{=} (\lambda + 1) \sum_{u=1}^{n} (n-1)[S(u) + 20]$.

Note that establishing the broadcast trees also involves some protocol overhead. We ignore such overhead since our focus is on the TCO during the state dissemination. For the proposed HFTB mechanism, no protocol overhead is incurred, since in HFTB the broadcast trees are established while LSAs are being flooded. Since HFTB floods the first LSA to determine the broadcast trees and then disseminates the subsequent LSAs over these trees, the TCO of HFTB is given by $\text{TCO}_{\text{HFTB}} \stackrel{\text{def}}{=} \sum_{u=1}^{n} m[S(u) + 20] + \lambda \sum_{u=1}^{n} (n-1)[S(u) + 20]$.

In S-HFTB, the complete LSAs are disseminated over the broadcast trees as in HFTB. However, to maintain consistencies in the broadcast trees, S-HFTB also disseminates 20-byte headers and their ACKs through neighbors that are not

children on the broadcast trees of the LSAs' originators. So these headers and their ACKs (total 40 bytes) are disseminated over $m - n + 1$ links, since we have $n - 1$ links in each tree. Hence, the TCO of S-HFTB can be computed as $\text{TCO}_{\text{S-HFTB}} \stackrel{\text{def}}{=} \sum_{u=1}^{n} m[S(u)+20]+\lambda\sum_{u=1}^{n}(n-1)[S(u)+20]+\lambda 40n(m-n+1)$.

We now compare the above four link-state dissemination mechanisms in terms of the TCO. We also report how much TCO reduction is achieved over flooding when HFTB, S-HFTB, or pure tree-based broadcasting is used. For HFTB and S-HFTB, the reduction in TCO over flooding is given by $100(1 - \frac{\text{TCO}_{\text{(S-)HFTB}}}{\text{TCO}_{flooding}})\%$. For tree-based broadcasting, the reduction is given by $100(1 - \frac{n-1}{m})\%$. The computation of the TCO depends on four parameters: n, m, q, and λ. For illustration purposes, we consider topologies with $n = 100$, and vary m, q, and λ. The same trends have been observed for other values of n. Figure 8 depicts the TCO of various dissemination mechanisms versus the number of links m. As m increases, the TCO of flooding increases quadratically, while this increase is linear in other mechanisms. Figure 9 considers the effect of q. As shown in the figure, the TCO of all mechanisms increases linearly with q. However, the TCO of flooding increases at a higher rate than those of tree-based mechanisms. Figure 10 considers the effect of λ. The TCO of all mechanisms is linearly proportional to λ. Once again, the TCO of flooding increases at a much higher rate than that of other mechanisms.

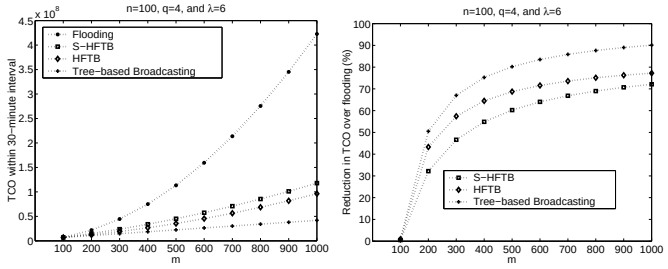

Fig. 8. Total communications overhead versus m.

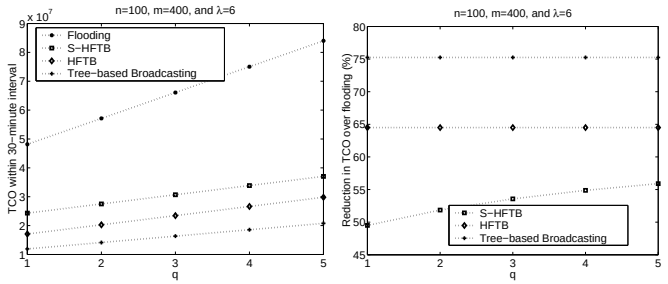

Fig. 9. Total communications overhead versus q.

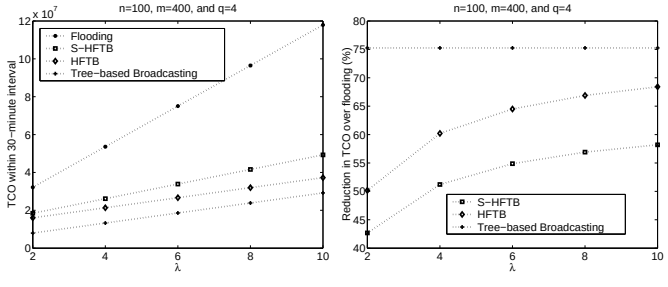

Fig. 10. Total communications overhead versus λ.

If the broadcast trees are already established, then the basic tree-based broadcasting provides the best possible efficiency in state dissemination. However, establishing such trees in the basic tree-based approach requires complex algorithms and protocols to maintain consistent trees throughout the network. Because of the complexities and/or unreliabilities of previous tree-based mechanisms, current Internet protocols do not use tree-based broadcasting, and instead rely on flooding despite its high communications overhead. The proposed (S-)HFTB mechanism takes advantage of both flooding and tree-based broadcasting, and provides significantly better performance than flooding. This is particularly advantageous in QoS routing, where the values of q and λ are typically larger than those in best-effort networks.

VI. Conclusions

We provided a hybrid state dissemination mechanism that combines tree-based broadcasting and flooding to achieve simple yet reliable and efficient link-state dissemination. Such a mechanism is particularly needed in the context of QoS routing, which involves frequent dissemination of several dynamic parameters. In contrast to previous tree-based broadcasting approaches, which require complex algorithms and protocols to determine and maintain the broadcast trees, the proposed HFTB simply determines the broadcast trees by flooding first-time LSAs. Subsequent LSAs that update the state of an existing link(s) are then disseminated over the broadcast trees. To deal with link-failures, we provided a modified version of HFTB, called S-HFTB. S-HFTB incurs a fixed extra communications overhead over HFTB. However, by using S-HFTB, nodes dynamically repair disconnected broadcast trees in the case of link failures and acquire the most recent LSAs in a simple and efficient manner. In spite of this overhead, the S-HFTB mechanism provides significantly better performance than currently used flooding while maintaining the simplicity and reliability of flooding. As a result, the S-HFTB approach is a viable alternative to standard flooding, with improved TCO. We also described how to integrate the proposed hybrid mechanism in OSPF. For this purpose, two of the currently unused bits of the Options field in the LSA header are defined as FT-bit and RT-bit. Using these bits, we slightly modified the flooding procedure of OSPF to determine the broadcast trees and to disseminate LSAs over these trees.

As a future work, we plan to implement the proposed hybrid mechanism along with OSPF and to demonstrate its efficiency in real systems.

References

[1] J. T. Moy, *OSPF: Anotomy of an Internet Routing Protocol*, Addison Wesley, 1998.

[2] J. Moy, "OSPF version 2," Standards Track RFC 2328, IETF, April 1998.

[3] The ATM Forum, "Private network-to-network interface specification version 1.0 (PNNI 1.0)," March 1996.

[4] G. Apostolopoulos, R. Guerin, S. Kamat, and S. K. Tripathi, "Quality of service based routing: A performance perspective," in *Proceedings of the ACM SIGCOMM '98 Conference*, Vancouver, British Columbia, Canada, August-September 1998, pp. 17–28.

[5] P. A. Humblet and S. R. Soloway, "Topology broadcast algorithms," *Computer Networks and ISDN Systems*, vol. 16, pp. 179–186, 1988/89.

[6] B. Bellur and R.G. Ogier, "A reliable, efficient topology broadcast protocol for dynamic networks," in *Proceedings of the INFOCOM '99 Conference*. IEEE, 1999, vol. 1, pp. 178 –186.

[7] G. Apostolopoulos, D. Williams, S. Kamat, A. Guerin, R. Orda, and T. Przygienda, "QoS routing mechanisms and OSPF extensions," RFC 2676, IETF, August 1999.

[8] G. Apostolopoulos, R. Guerin, and S. Kamat, "Implementation and performance measurements of QoS routing extensions to OSPF," in *Proceedings of the INFOCOM '99 Conference*. IEEE, 1999, vol. 2, pp. 680 –688.

[9] E. Basturk and P. Stirpe, "A hybrid spanning tree algorithm for efficient topology distribution in PNNI," in *Proceedings of the 1st IEEE International Conference on ATM (ICATM '98)*, 1998, pp. 385–394.

[10] Y. K. Dalal and R. M. Metcalfe, "Reverse path forwarding of broadcast packets," *Communications of the ACM*, vol. 21, pp. 1040–1048, December 1978.

[11] T. H. Cormen, C. E. Leiserson, and R. L. Rivest, *Introduction to Algorithms*, The MIT press and McGraw-Hill book company, sixteenth edition, 1996.