Title	Link layer multi-priority frame forwarding
Author(s)	Lui, KS; Lee, WC; Nahrstedt, K
Citation	Ieee International Conference On Communications, 2003, v. 3, p. 1573-1577
Issued Date	2003
URL	http://hdl.handle.net/10722/46346
Rights	Creative Commons: Attribution 3.0 Hong Kong License

Link Layer Multi-Priority Frame Forwarding

King-Shan Lui
Department of Electrical and Electronic Engineering
University of Hong Kong
Pokfulam Road, Hong Kong
Email: kslui@eee.hku.hk

Whay Chiou Lee
Broadband Networks Research Lab
Motorola Labs
Mansfield, MA 02048, USA
Email: Whay.Lee@motorola.com

Klara Nahrstedt
Department of Computer Science
University of Illinois at Urbana-Champaign
Urbana, IL 61801, USA
Email: klara@cs.uiuc.edu

***Abstract*— With increasing demand for multimedia and real-time applications, local area network (LAN) technologies are rapidly being upgraded to support Quality-of-Service (QoS). Many QoS-enabled LANs are making use of resource allocation mechanisms that can discriminate among traffic classes of different priorities. When such LANs are interconnected by bridges to form an extended LAN, it is necessary to upgrade the bridges so that they are QoS-enabled as well. For example, the IEEE 802.1p standard defines a framework for priority queuing in bridges. Alternatively, frame forwarding decisions at the link layer may be modified to recognize frame priorities and alternate paths may be used for differentiating QoS. In this paper, we describe a novel bridge protocol that can forward frames of different priorities using different paths. Our protocol ensures that the forwarding path of a higher priority frame is never longer than the forwarding path of a lower priority frame.**

I. Introduction

With increasing demand for multimedia and real-time applications, local area network (LAN) technologies are rapidly being upgraded to provide support for quality-of-service (QoS). To this end, traffic prioritization at the link layer has become common practice. In an extended LAN, where multiple LAN segments are interconnected via bridges, a frame sent from one LAN to another may go through one or more bridges. A need thus arises for frame forwarding over alternate paths based on priorities.

Bridges operate on top of the *Medium Access Control* (MAC) layer, which is a sublayer of the data link layer. The data unit in this layer is called *frame* or *MAC frame*. *MAC addresses* are used to identify hosts. The basic function of a bridge is to forward *MAC frames* from one LAN to another without requiring any modification to the communication software in the hosts. Bridges do not modify the content or format of the MAC frames they receive and the operation of bridges should not misorder or duplicate frames.

In the standard IEEE 802.1D spanning tree bridge protocol, a shortest path spanning tree with its root at a predetermined bridge, known as a *root bridge*, is used to interconnect LANs together. The spanning tree defines a unique path between each pair of bridges/LANs, and the same path is used to forward all frames transported between the pair. As only one spanning tree is used, the forwarding path between a pair of LANs may not be a shortest path. Moreover, traffic is restricted on a single spanning tree and some links are not used for frame forwarding at all. To enhance the routing capability of the standard bridge protocol, many enhancements to the standard have been proposed. Most of them focus on finding an alternate path that is shorter than the corresponding tree path between a pair of bridges/LANs.

To provide QoS in extended LANs, the IEEE 802.1 group has developed enhancements to bridge functions. In particular, the IEEE 802.1p standard, which is incorporated in the most recent IEEE 802.1D standard [1], defines a priority queuing framework for 802.1D bridges. Another IEEE standard, namely 802.1Q [2], specifies a frame format for carrying such priority information. Each frame associated with a priority is queued according to that priority value. The default scheduling method is to send frames in a lower priority queue only when there is no frame waiting in a higher priority queue. There are at most eight different user priorities but the number of different queues a bridge has may vary. In order to avoid frame misordering, the mapping between priorities and queues is static. The IEEE standard does not define how a host assigns user priorities to frames. IETF, on the other hand, has studied how to support integrated service over extended LANs that supports the IEEE 802.1D/p standard [3], [4], [5] .

Although frames of different priorities may be put in different queues, they all go through the same forwarding path if they go from the same source to the same destination. Therefore, if a link is very congested, the highest priority frames will also suffer high delay, while the lower priority frames will suffer severe starvation. In this respect, it is desirable to differentiate the forwarding paths of frames of different priorities. To the best of our knowledge, none of the existing mechanisms forwards frames with different priorities using different paths. In this paper, we describe a novel bridge

This work was supported by Motorola Inc. through the Broadband Networks Research Lab and the Motorola Center for Communications at the University of Illinois at Urbana-Champaign.

0-7803-7802-4/03/$17.00 © 2003 IEEE

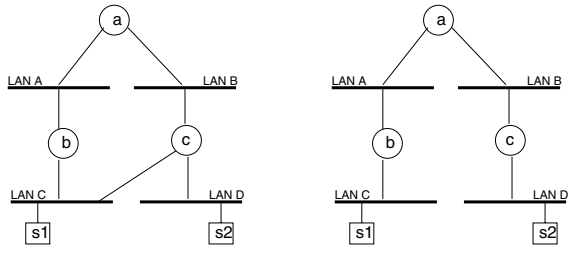
protocol that is able to forward frames using different alternate paths based on frame priorities. Our protocol, which can interoperate with the standard IEEE 802.1D bridge protocol and some existing extensions, ensures that a frame of given priority always goes through a forwarding path that is no worse than a forwarding path used by any lower-priority frame sent from the same source host to the same destination host. The storage and computation required by the protocol are asymptotically the same as that of the standard bridge protocol. Moreover, our protocol is very flexible that the differentiation pattern can be adjusted according to user needs.

In the rest of this paper, we describe the standard protocol and its extensions in Section II. We describe our protocol in Sections III and Section IV. We discuss the performance of our protocol in Section V and draw conclusion in Section VI.

II. Standard Protocol and its Extensions

Each IEEE 802.1D bridge has three basic functions: (1) frame forwarding, (2) learning, and (3) spanning tree construction. (1) and (2) are performed with the use of a *Forwarding Database* (FD) within each bridge. An FD in a bridge specifies which port of the bridge to forward a data frame, when a destination host is given. If there is no such entry in the FD, the bridge forwards the frame through all ports except the port through which the frame came. Whenever a frame from source host s is received at port p, the bridge marks in its FD that the forwarding port of s is p. Entries in the FD are forgotten periodically.

Because a bridge forwards a frame with unknown destination to all ports except the incoming one, if there are loops in the bridged LAN, then, a frame may be forwarded indefinitely. To avoid this, a distributed spanning tree algorithm is used to make sure that the active topology among the bridges is always a tree so that there is a unique path between each pair of bridges/LANs/hosts. The bridge with the smallest identifier is selected to be the root and a shortest path tree w.r.t. the root is built by exchanging configuration messages. Links that are not selected to be tree links, called *non-tree links* will be disabled and never used to forward frames. As only one tree is used, the forwarding path between two hosts may not be a shortest path.



(a) Extended LAN (b) Spanning Tree

Fig. 1. Extended LAN with hosts $s1$ and $s2$

Figure 1(a) shows a simple extended LAN. a, b, and c are bridges, while LAN A, LAN B, LAN C, and LAN D are four different LANs. $s1$ is a host on LAN C and $s2$ is a host on LAN D. There is a loop in this extended LAN. To avoid indefinite frame forwarding, the standard protocol will deactivate some of the links to form a spanning tree so that there is a unique path between each pair of bridges. Let a be the root bridge. The spanning tree, built according to hop count, is shown in Figure 1(b). The link between bridge c and LAN C is deactivated. Suppose that host $s1$ on LAN C wants to send a frame to another host $s2$ which is on LAN D. Although both b and c connect to LAN C, only b will process the frame, since the port where c connects to LAN C is disabled by the spanning tree algorithm. Therefore, when $s1$ sends a frame to $s2$, the path of the frame is $s1 \rightarrow b \rightarrow a \rightarrow c \rightarrow s2$. This tree path is longer than the shortest path, $s1 \rightarrow c \rightarrow s2$ from LAN C and LAN D.

To improve the routing capability of the standard, several extensions have been proposed. Most of them focus on finding shorter forwarding paths [6], [7], [8], [9], [10], [11], [12]. All frames that have the same source and destination are forwarded over the same path, despite of the priority. Hart proposed to forward frames over non-tree link to achieve load sharing [13], [14]. However, there is no guarantee that the non-tree link is better than the tree path and so it is difficult to determine on which path a higher priority frame should be forwarded on. Moreover, a non-tree link is used only for the traffic between the end bridges of that link. To the best of our knowledge, there is no existing protocol that can forward frames according to their priorities on more than two paths. Our protocol utilizes the capability of finding shorter alternate paths provided by [6] - [12] to forward frames on different paths. By forwarding frames on different paths, our protocol diverts traffic onto non-tree links and achieves better load balancing. Our protocol is very flexible that it can work with different existing extensions of the standard protocol and the pattern of differentiation is adjustable.

III. Protocol Overview

A. Model

Our protocol utilizes the capability of finding shorter alternate paths provided by [6] - [12]. As different protocols use different models, we follow the model used in [12] to illustrate the general idea. In [12], shorter paths among bridges are found for frame forwarding and hosts are mapped to the bridges to utilize the shorter forwarding paths. Therefore, in this paper, we describe how to forward frames on different paths between a pair of bridges.

We represent a bridged LAN as an undirected graph where bridges are nodes. A link connects two bridges together. Each link is associated with a non-negative cost. The length of a path is the sum of the costs of all the links along the path. The priority of a frame is an integer between 0 to 7. A smaller number implies a higher priority and a frame of priority value 0 is of the highest priority. The problem of multi-priority frame forwarding is modeled as message forwarding from one node to another according to the priority of the messages.

B. Overview

The standard forwards frames on tree paths only and the extensions forward frames on at most two different paths: the

tree path and the shortest alternate path identified. Apart from the tree path and the shortest alternate path, our protocol can forward frames, according to their priorities, over one or more *hybrid paths*, where each consists of a segment of a tree path, followed by a shortest alternate path. Figure 2 illustrates a hybrid path from z to v. In the following figures in this paper, hop count is the considered additive metric, solid links are tree links, and dash links are non-tree links. The tree path from z to v is $z \rightarrow y \rightarrow x \rightarrow q \rightarrow u \rightarrow v$ and the shortest alternate path from y to v is $y \rightarrow u \rightarrow v$. The hybrid path from z to v consists of a segment of the tree path from z to v ($z \rightarrow y$) and the shortest path from y to v. The length of a hybrid path is less than that of the tree path and is larger than the length of the shortest alternate path identified. Therefore, our protocol would forward frames of intermediate priorities over hybrid paths and we will describe the detail in Section IV.

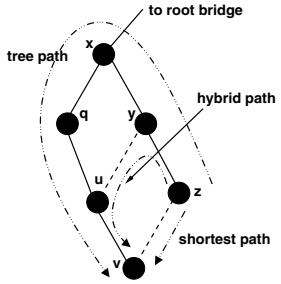


Fig. 2. Hybrid Path

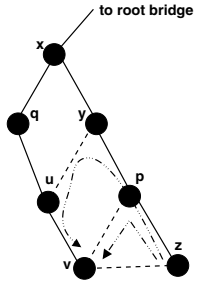


Fig. 3. Multiple Hybrid Paths

C. Information Required

To forward frames on hybrid paths, our protocol requires each node to keep the information of two forwarding paths to each other node: the tree path and the shortest path. Refer to Figure 2, z only has to keep the next hop node on the tree path to v, which is y, and the next hop node on the shortest path to v, which is v itself. Therefore, the storage requirement of our protocol is asymptotically the same as existing protocols. The next hop of the shortest path can be obtained by the extensions described in Section II. The tree path forwarding direction is in the FD specified by the standard. Therefore, there is little extra overhead in finding the two forwarding directions.

Apart from forwarding directions, our protocol requires each node to keep track of the number of nodes on the tree path to every other node. We will describe why we need this information when we describe our protocol in detail in Section IV. To obtain the number of nodes on a tree path, After the standard spanning tree is built, each node sends a frame called `Tree_Hop_Count` on all of its tree links. This frame contains a counter that reflects how many nodes the frame has passed through and the last node that processes that frame. The node that originates the frame initializes the counter to 0. When a node x receives a `Tree_Hop_Count` with counter value c that is originated from another node y, x knows that y is c hops away on the tree path, denoted as $tree\_hop\_count(x, y) = c$. x then increments the counter by 1 and sends the frame on all tree links except the link that `Tree_Hop_Count` comes from.

The propagation of a `Tree_Hop_Count` frame is the same as the propagation of a data frame which is sent from an unknown host. Therefore, the complexity of this step is the same as sending data frames to several unknown hosts in the standard protocol. Note that in an extended LAN, hosts must outnumber bridges. Moreover, as the entries in the FD are forgotten periodically, hosts would become unknown after idling for certain time. As a result, the message overhead of `Tree_Hop_Count` is negligible.

IV. Multi-Priority Frame Forwarding

In this section, we discuss how a hybrid path is formed and present the complete protocol. To simplify our discussion, we denote a tree path from node x to node y as $treepath(x, y)$, the shortest alternate path identified between x and y as $shortpath(x, y)$, and the hybrid path as $hybrid(x, y)$.

A. Hybrid Paths and Frame Forwarding

$hybrid(z, v)$ in Figure 2 consists of $treepath(z, y)$, which is a segment of $treepath(z, v)$, and $shortpath(y, v)$. A frame would go through this path because (1) z forwards the frame over $treepath(z, v)$ towards y, and (2) y forwards the frame over $shortpath(y, v)$. In other words, when an intermediate node on the tree path diverts the frame, which is being forwarded on the tree path, to the shortest alternated path identified from itself to the destination, the frame will be forwarded on a hybrid path. Therefore, when a node receives a frame from downstream, it has to decide whether to forward it over the tree path continuously, or to forward it over the shortest alternate path identified. It must determine the forwarding direction in such a way that the forwarding path of a higher priority frame is no worse than the forwarding path of a lower priority frame that is sent from the same source to the same destination.

We now consider two different hybrid paths from z to v as shown in Figure 3. Hybrid path $P1 = z \rightarrow p \rightarrow v$ is shorter than hybrid path $P2 = z \rightarrow p \rightarrow y \rightarrow u \rightarrow v$. In $P1$, the tree path segment is $treepath(z, p)$ and in $P2$, the tree path segment is $treepath(z, y)$. That is, a hybrid path that traverses fewer tree hops is shorter. The key idea of our protocol is: the higher the priority of a frame, the smaller the number of tree hops it has to traverse. In addition, for a frame of a given priority, the more tree hops it has made, the more likely it should be forwarded over an available alternate path. Refer to Figure 3, a frame of the highest priority should be forwarded using the non-tree link (z, v), without going through any tree hop. For a frame of the next lower priority, z should forward it to p, using the tree path forwarding direction and let p forward it over $shortpath(p, v)$. A frame of the next lower priority is forwarded on tree forwarding port by z and p to y and y forwards it over $shortpath(y, v)$. Finally, a frame of the lowest priority is forwarded over the tree path, that is, all nodes decide to send it to the next hop on the tree path.

B. Forwarding Decision Function

A node uses the priority of the frame and the number of tree hops that the frame has traversed to determine where to forward

the frame. A node invokes a function to make the forwarding decision. We call this function *forwarding decision function*. This function takes two values: (1) the priority of the frame ($priority\_value$) and (2) the number of tree hops traversed (hop_count), and returns the forwarding direction of the frame. There are at most two different forwarding directions: the next hop on the tree path, or the next hop on the shortest alternate path identified. We denote these two directions as $next\_tree\_hop$ and $next\_short\_hop$. If the function returns a positive value, the forwarding direction is $next\_tree\_hop$; otherwise, the forwarding direction is the $next\_short\_hop$.

Generally speaking, as long as a function $m(priority\_value, hop\_count)$ satisfies the following conditions, it can be used as the forwarding decision function:

1) $m(p, h1) \leq m(p, h2)$ if $h1 \geq h2$
2) $m(p1, h) \geq m(p2, h)$ if $p1 \geq p2$

Condition (1) says that the more tree hops a frame has traversed, the more likely that it is forwarded using a shorter path. Condition (2) says that a frame of higher priority (smaller number means a higher priority) is more likely to be forwarded on a shorter path.

Different functions would yield different forwarding decisions for the same frame. For example, the function $priority\_value - hop\_count$ will forward frames of priority 0 to $next\_short\_hop$ while other frames over $next\_tree\_hop$, if the node that executes the function is the source node. On the other hand, if the function is $\lfloor \frac{priority\_value}{4} \rfloor - hop\_count$, a source node will forward frames of priority values 0 - 3 to $next\_short\_hop$. Hence, the forwarding decision function can be adjusted for different forwarding patterns, making our protocol very flexible.

C. Complete Protocol

When a node n receives a frame that is sent from source s to destination t with priority value $priority\_value$, it checks whether $treepath(n, t)$ is the same as $shortpath(n, t)$. If so, it sends the frame towards $next\_tree\_hop$; otherwise, it computes $m(priority\_value, tree\_hop\_count(n, s))$. If the forwarding decision function returns a positive number, n sends the frame towards $next\_tree\_hop$. If the forwarding decision function returns zero or a negative number, n sends the frame towards $next\_short\_hop$.

Consider a frame that is sent from z to v with priority value 1 in Figure 3. Suppose that the forwarding decision function is $priority\_value - tree\_hop\_count$. When z processes the frame, the forwarding decision function returns 1 - 0 = 1 $>$ 0. Therefore, z sends it to $next\_tree\_hop$, which is p. When p receives the frame, as $tree\_hop\_count(p, z) = 1$, the forwarding decision function returns 0 and p forwards the frame to its $next\_short\_hop$, which is v, the destination. The path that the frame goes through is $z \rightarrow p \rightarrow v$.

For a frame that is sent from z to v with priority 2, z will send it to p as in the case of the frame with priority 1. When p computes the forwarding decision function, the function returns 2 - 1 = 1 and so p forwards the frame to y. When y receives the frame, the forwarding decision function returns 0 and then y forwards it to $next\_short\_hop$, which is u. As $treepath(u, v)$ is the same as $shortpath(u, v)$, u forwards the frame to v using the tree link. The path that this frame traverses is $z \rightarrow p \rightarrow y \rightarrow u \rightarrow v$.

V. Performance Analysis

A. Correctness

Let us assume that two frames, $f1$ and $f2$, are being sent from the same source s to the same destination t. Suppose that the priority values of them are $p1$ and $p2$, respectively. Our protocol ensures that, without loss of generality, if $p1 < p2$, the forwarding path of $f1$ is no longer than the forwarding path of $f2$.

To see this, suppose that the forwarding path of $f2$ is composed of $treepath(s, n)$ and $shortpath(n, t)$. That is, all the nodes on $treepath(s, n)$, except n, forward $f2$ to $next\_tree\_hop$, while n forwards $f2$ to $next\_short\_hop$. We further assume that the forwarding path of $f1$ consists of $treepath(s, n')$ and $shortpath(n', t)$. If $n = n'$, our claim holds. If $n \neq n'$ and n' is on $treepath(s, n)$, the forwarding path of $f1$ is shorter than the one of $f2$. If $n \neq n'$ and n' is not on $treepath(s, n)$, n must receive $f1$. According to Condition (2) of the forwarding decision function, if n decides to forward $f2$ on $shortpath(n, t)$, it must also decide to forward $f1$ on $shortpath(n, t)$. Therefore, the forwarding path of $f1$ must be shorter than or the same as the forwarding path of $f2$. A formal proof can be found in [15].

B. Number of Different Paths

For nodes that are on the same branch, that is, one is an ancestor of the other on the tree, the tree path is the shortest path between them and there is no path differentiation. There are different paths only when the nodes are on different branches and shorter alternate path exists, such as z and v in Figure 3. The tree path between nodes on different branches consists of an upstream segment from the source to the nearest common ancestor of the nodes ($treepath(z, x)$) and a downstream segment from the nearest common ancestor to the destination ($treepath(x, v)$). Among the nodes n on $treepath(z, v)$, it is possible for $treepath(n, v)$ to be different from $shortpath(n, v)$ only when n is on the upstream segment but not the nearest common ancestor. Refer to Figure 3, the shortest path is different from the tree path for nodes y, p, and z to node v. However, x, which is an ancestor of v, does not have a non-tree path to v that is shorter than $treepath(x, v)$. Therefore, the maximum number of different paths from node s to node t depends on the number of nodes on the upstream tree path from s to t.

C. Simulation Results

We conducted simulation studies to evaluate the performance of our protocol. We generated a total of 2200 networks of different topologies. Each of these networks was generated in three steps. In the first step, a rooted spanning tree was generated, for a fixed network size, by randomly varying branching factors at each node, wherein a branching factor at

a node refers to the number of children of the node with respect to the rooted spanning tree to be generated. In the second step, non-tree links were added to connect some pairs of nodes that were not immediate neighbors on the rooted spanning tree. In the third step, a random cost was assigned to each link.

The network size was varied from 20 to 30 nodes, resulting in a total of 11 different network sizes. There were two different ranges for the branching factors, one for high branching factors and one for low branching factors. At the root, high branching factor varied from 6 to 8 while low branching factor were from 4 to 6. Ranges 2 to 4 and 4 to 6 were used for the branching factors at other nodes. 50 different topologies were randomly constructed for each combination of network size and branching factors. Each tree link cost was a random integer between 1 and 3 inclusive. For each node, both the number of non-tree neighbors and the non-tree neighbors were randomly selected. As a bridge has only a few outgoing interfaces, we restricted the number of non-tree links a node could have to be at most the network size divided by 7. The costs of non-tree links were set in a way that the tree structure generated in the first step is preserved in the computation of a rooted shortest path tree. In this respect, the cost of each non-tree link was set to a sufficiently large value and augmented randomly by 1 or 2.

We measured the average path length ratio (average $\frac{\text{length of tree path}}{\text{length of our forwarding path}}$) of three different priorities and the result is shown in Table I. We also studied the number of different paths available between a pair of nodes (Table II). The results show that our protocol successfully forward frames on two or more paths according to frame priorities. In this respect, our protocol is expected to perform better than existing bridge protocols which are not equipped to do the same. Moreover, such an advantage of our protocol will be greater in large networks where there are many alternate paths for prioritized frame forwarding.

Network Size	1st priority	2nd priority	3rd priority
20	1.1281	1.0374	1.0006
21	1.2270	1.0656	1.0005
22	1.2289	1.0714	1.0019
23	1.2251	1.0701	1.0019
24	1.2262	1.0708	1.0030
25	1.2281	1.0719	1.0039
26	1.2322	1.0760	1.0044
27	1.2401	1.0809	1.0047
28	1.3256	1.1086	1.0092
29	1.3075	1.1004	1.0081
30	1.3157	1.1074	1.0093

TABLE I
PATH PERFORMANCE OF DIFFERENT PRIORITIES

Size	Avg. Number of Paths	Max. Number of Paths
20	1.2260	2.9850
21	1.4004	3.0200
22	1.4090	3.1100
23	1.4093	3.1050
24	1.4122	3.1750
25	1.4252	3.2250
26	1.4444	3.2750
27	1.4616	3.3150
28	1.6112	3.4350
29	1.5968	3.4600
30	1.6211	3.4900

TABLE II
NUMBER OF PATHS AVAILABLE

VI. CONCLUSION

In this paper, we describe a novel bridge protocol, which can forward frames using different paths according to their frame priorities. Our protocol ensures that the forwarding path of a frame of a given priority is never worse than that of any lower priority frame, provided that the frames are sent from the same source to the same destination. Our protocol can work upon any existing bridge protocol that tries to find shorter forwarding paths than the standard does without increasing the asymptotic complexity of the protocol. In the future, we would like to study the effect of different forwarding decision functions and explore the use of dynamic forwarding functions to achieve dynamic load balancing.

REFERENCES

[1] *Information technology - telecommunications and information exchange between systems - local and metropolitan area networks - common specifications. Part 3: Media Access Control (MAC) bridges, ISO/IEC 15802-3, ANSI/IEEE Std 802.1D*, 1998.

[2] *IEEE Standards for Local and Metropolitan Area Networks: Virtual Bridged Local Area Networks, IEEE Std 802.1Q*, 1998.

[3] A. Ghanwani, W. Pace, V. Srinivasan, A. Smith, and M. Seaman, "A Framework for Providing Integrated Services Over Shared and Switched LAN Technologies," RFC 2816, 2000.

[4] R. Yavatkar, D. Hoffman, Y. Bernet, F. Baker, and M. Speer, "SBM (Subnet Bandwidth Manager): A Protocol for Admission Control over IEEE 802-style Networks," RFC 2814, May 2000.

[5] M. Seaman, A. Smith, E. Crawley, and J. Wroclawski, "Integrated Service Mappings on IEEE 802 Networks," RFC 2815, 2000.

[6] R. Perlman et. al., "Utilization of Redundant Links in Bridged Networks," U.S. Patent Number 5,150,360, Sept. 22, 1992.

[7] B. Rajagopalan and M. Faiman, "Load Sharing and Shortest-Path Routing in Transparently Interconnected Local Area Networks," in *INFOCOM*, 1991.

[8] Y.-D. Lin and M. Gerla, "Brouter: The Transparent Bridge with Shortest Path in Interconnected LANs," in *LCN*, 1991.

[9] T.-Y. Tai and M. Gerla, "LAN Interconnection: A Transparent, Shortest-Path Approach," in *ICC '91*, 1991.

[10] R. Garcia, J. Duato, and J.J. Serrano, "A New Transparent Bridge Protocol for LAN Internetworking Using Topologies with Active Loops," in *International Conference on Parallel Processing*, 1998.

[11] T. Rodeheffer, C. Thekkath, and D. Anderson, "SmartBridge: A Scalable Bridge Architecture," in *SIGCOMM*, 2000.

[12] K. Lui, W. Lee, and K. Nahrstedt, "STAR: A Transparent Spanning Tree Bridge Protocol with Alternate Routing," *ACM Computer Communication Review*, vol. 32, no. 3, July 2002.

[13] J. Hart, "Extending the IEEE 802.1 MAC Bridge Standard to Remote Bridges," *IEEE Network Magazine*, vol. 2, no. 1, Jan. 1988.

[14] J. Hart, "Distributed Load Sharing," U.S. Patent Number 4,811,337, Mar. 7, 1989.

[15] K. Lui, *Alternate Routing Protocols for Bridged Networks*, Ph.D. thesis, University of Illinois, Urbana-Champaign, May 2002.