STAR: A Transparent Spanning Tree Bridge Protocol with Alternate Routing *

King-Shan Lui
Dept. of Computer Science
University of Illinois at Urbana-Champaign
kinglui@cs.uiuc.edu

Whay Chiou Lee
Broadband Networks Research Lab
Motorola Labs
Whay.Lee@motorola.com

Klara Nahrstedt
Dept. of Computer Science
University of Illinois at Urbana-Champaign
klara@cs.uiuc.edu

ABSTRACT

With increasing demand for multimedia applications, local area network (LAN) technologies are rapidly being upgraded to provide support for quality of service (QoS). In a network that consists of an interconnection of multiple LANs via bridges, the QoS of a flow depends on the length of an end-to-end forwarding path. In the IEEE 802.1D standard for bridges, a spanning tree is built among the bridges for loop-free frame forwarding. Albeit simple, this approach does not support all-pair shortest paths. In this paper, we present a novel bridge protocol, the Spanning Tree Alternate Routing (STAR) Bridge Protocol, that attempts to find and forward frames over alternate paths that are provably shorter than their corresponding tree paths. Being backward compatible to IEEE 802.1D, our bridge protocol allows cost-effective performance enhancement of an existing extended LAN by incrementally replacing a few bridges in the extended LAN by the new STAR bridges. We develop a strategy to ascertain bridge locations for maximum performance gain. Our study shows that we can significantly improve the end-to-end performance when deploying our bridge protocol.

1. INTRODUCTION

Bridges are devices used to interconnect several local area networks (LANs) to form an extended LAN. 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. A bridge has several ports connecting different LANs. A frame sent from one LAN to another will typically go through one or more bridges. This bridged LAN environment should be transparent to hosts and should look like a single LAN to the hosts. The basic function of bridges 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. More detail of bridges can be found in [13, 14, 18, 4, 5].

IEEE 802.1D Spanning Tree Bridge Protocol is a widely used standard for interconnecting the family of IEEE 802 standard LANs [3]. In this standard, a shortest path spanning tree with its root at a predetermined bridge, known as a *root bridge*, is used to interconnect LANs to form an extended LAN. The spanning tree defines a unique path between each pair of LANs, but this path may not be a shortest path. Moreover, as only one spanning tree is used, some bridges and some ports may not be used at all. As the bridge protocol is now adopted not only in extended LANs, but also in cable networks [1], home networks [17, 22], and metropolitan area networks [2], future bridged networks will span a wide range of sizes and geographical coverage. Therefore, to support multimedia and real time applications with stringent quality of service (QoS) requirements in these kinds of networks, the QoS routing capability of bridges has to be enhanced.

We propose a novel bridge protocol that finds and forwards frames over alternate paths if possible. This proposed protocol, referred to as *Spanning Tree Alternate Routing* (STAR) Bridge Protocol, has a complexity that is comparable to that of the standard and other existing protocols. We have required the STAR Bridge Protocol to be backward compatible with the standard bridge protocol, so that it is possible to incrementally upgrade standard-based bridges in an extended LAN to be STAR bridges to take advantage of shorter alternate paths. We have thus devised a bridge replacement strategy that aims at maximizing performance gain, subject to a limit on the number of bridges replaced.

In the rest of this paper, we describe the IEEE 802.1D Spanning Tree Bridge Protocol in Section 2 and related work in Section 3. We present the STAR Bridge Protocol in Section 4 and the replacement strategy in Section 5. We analyze the performance and complexity of the protocol in Section 6. Finally, we conclude in Section 7.

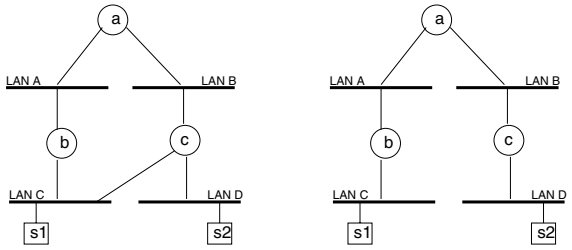
2. IEEE 802.1D SPANNING TREE BRIDGE PROTOCOL

Each IEEE 802.1D bridge has three basic functions: (1) frame forwarding, (2) learning, and (3) spanning tree construction. Functions (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 to a particular destination station. 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 station s is received at port

*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.

p, the bridge marks in its FD that the forwarding port of s is p. If there are loops in the bridged LAN, a frame may be forwarded indefinitely. To avoid this, function (3) is used to make sure the active topology among the bridges is always a tree so that there is a unique path between each pair of bridges. We refer to such a path as a *tree path*.

A distributed spanning tree algorithm is used to construct a shortest path tree rooted at a selected bridge known as a root bridge. This root bridge is selected using bridge identifiers. A path that connects a bridge and the root bridge over the spanning tree is referred to as a *root path* associated with that bridge. *Non-tree links* are links that have not been selected by the 802.1D spanning tree algorithm. By exchanging configuration messages, bridges identify the root bridge and select which ports to activate. For each LAN, a single bridge is elected among all bridges connected to the LAN to be the *designated bridge*. This designated bridge is the closest bridge to the root bridge. In order to maintain an up-to-date tree that reflects the underlying topology, the root bridge broadcasts configuration messages periodically over the spanning tree to all other bridges. Whenever a topological change is detected, the bridges will start building a new tree. Data frames will not be forwarded until the new tree is built.



(a) Extended LAN (b) Spanning Tree

Figure 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. Let a be the root bridge. The spanning tree, built according to hop count, is shown in Figure 1(b). The designated bridge of LAN D is c and the designated bridge of LAN C is b. 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.

3. RELATED WORK

The IEEE 802.1D Spanning Tree Bridge Protocol has previously been extended and modified in a variety of ways to improve its routing capability. To the best of our knowledge, our protocol is the first protocol that can ensure that a non-tree forwarding path is no worse than the corresponding tree path for any arbitrary additive metric in an extended LAN where there are standard bridges.

Some prior methods augment the spanning tree by allowing non-tree links to be additionally used for frame forwarding under appropriate conditions [8, 9, 6]. The method described in [8] and [9] only ensures that the length associated with a selected non-tree link for frame forwarding is no greater than the sum of the root path distances associated with the two bridges at the ends of the selected non-tree link. Hence, a selected alternate forwarding path may be longer than its corresponding tree path, as in the case where the two root paths share a common path segment. This method is extended in [6] to enable the "speed" of a non-tree link to be compared to that of its corresponding tree path. The "speed" is determined by having a bridge on one end of the non-tree link send to the bridge on the other end of the non-tree link a special message over the non-tree link or the corresponding tree path. This method cannot guarantee that a forwarding path is no worse than its corresponding tree path for any additive metric considered except when the additive metric is derived from "speed."

The method described in [15] dynamically creates a shortest path tree rooted at a given source host from a default spanning tree by activating some non-tree links and disabling some tree links on demand according to a delay measure. In methods described in [11] and [20], distance vectors are maintained in bridges showing the shortest path direction for forwarding frames to a particular LAN, and not to a station. Mapping tables, which are used to map stations to LANs, are exchanged by means of flooding. In [16], a bridge architecture with IP routing capability is proposed, wherein topology information is exchanged among bridges to enable a shortest path to every LAN to be found. The architecture also has a mechanism to locate end stations. None of the methods described in [15, 11, 20, 16] are backward compatible with the IEEE 802.1D Spanning Tree Bridge Protocol.

The method described in [7] enables optimal or sub-optimal paths to end stations to be identified using a distance vector approach. Although this method is backward compatible with the standard spanning tree bridge protocol, it is possible for a path determined by this method to be longer than its corresponding tree path when there are bridges that do not execute this method in the extended LAN.

4. SPANNING TREE ALTERNATE ROUTING BRIDGE PROTOCOL

We describe the STAR Bridge Protocol in this section. Bridges that deploy the proposed protocol are referred to as *STAR bridges* and those that execute the standard protocol only are called *STD bridges*. There are two main design goals in this protocol: enhanced forwarding path performance and backward compatibility. STAR bridges attempt to forward frames over alternate paths that are shorter than their corresponding tree paths on the standard spanning tree, provided that such alternate paths can be identified without excessive protocol complexity. The metric used may be delay, cost, or any other additive metric. By appropriate placement of STAR bridges in an extended LAN, the average forwarding path length may be reduced even if only a few STAR bridges are used. As current extended LANs are large [10], it can be costly to replace all STD bridges by STAR bridges. We will describe a strategy to find good replacement locations in Section 5. By being backward compatible with the IEEE 802.1D standard, STAR bridges operate seamlessly with STD bridges.

In this section, we first give an overview of the STAR bridge protocol (see Section 4.1) and describe the model (see Section 4.2). We then discuss the details of the protocol in Sections 4.3 to 4.5.

4.1 Overview

In the IEEE 802.1D standard bridge protocol, there is a separate process responsible for each basic function specified in Section 2. In our STAR Bridge Protocol, three new processes are further specified. They are *STAR path finding process*, *STAR learning process*, and *STAR forwarding process*. The path finding process allows a STAR bridge to find and estimate the distance of a path from itself to another STAR bridge. The STAR forwarding process and the STAR learning process are modified versions of the forwarding process and the learning process specified in the standard respectively. All STAR bridges can execute both the standard and the new processes.

Figure 2 is a state diagram of the STAR bridge protocol. In the standard, a *rooted spanning tree* (RST) is built by means of a distributed algorithm before the forwarding and learning processes start. A timeout mechanism is used as an indication of the completion of the RST construction. In our protocol, an RST is found by STD and STAR bridges together before the execution of the new processes. After the RST is found, the path finding process is started. Before the path finding process ends, STAR bridges and STD bridges execute the standard forwarding process and the standard learning process to forward data frames on tree paths. When the path finding process ends, each STAR bridge begins to execute the STAR learning process and the STAR forwarding process instead of the standard ones to forward data frames on identified enhanced forwarding paths. We adopt a timeout mechanism to indicate the end of the path finding process. We use the same timer time as the one used in the RST construction.

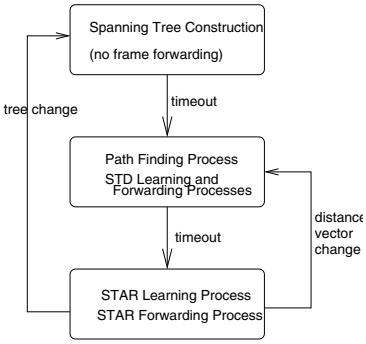


Figure 2: State Diagram of STAR Bridge Protocol

In the IEEE 802.1D standard, each bridge keeps an FD for the forwarding process. In STAR bridges, two tables are additionally used: *bridge forwarding table* (BF Table) and *host location table* (HL Table). A BF Table indicates the forwarding port to each other STAR bridge in the bridged LAN along the "best" path found. BF Tables are obtained in the path finding process by a modified distance vector method as described in Section 4.3. Since the BF Table contains forwarding information to STAR bridges only, it is not sufficient to forward data frames which are destined to hosts, not bridges. Therefore, an HL Table is used to map a host to a STAR bridge near it. The STAR learning process is responsible for filling up this table. We will describe the STAR processes in Section 4.3 through Section 4.5.

4.2 Model

We represent a bridged LAN as an undirected graph G = (B, D, E), where B is the set of STAR bridges, D is the set of STD bridges. and E be the set of links connecting the bridges. Each link $(x, y) \in E$ is assumed to have a non-negative cost $c(x, y)$. For convenience, we let $c(x, y) = \infty$ if $(x, y) \notin E$. If there are several links between bridge x and bridge y, $c(x, y)$ should be the minimum among the costs of the links. A path in G is a loop-free tandem concatenation of links in E. The length of a path is the sum of the costs of all the links along the path. For example, if the metric being considered is hop count, $c(x, y)$ for each $(x, y) \in E$ would be 1. Table 1 summarizes the notations used in this section.

Notation	Meaning
G	bridged LAN graph
B	set of STAR bridges in G
D	set of STD bridges in G
V	set of all bridges in G, $V = B \cup D$
E	set of links in G
(x, y)	a link from bridge x to bridge y
$c(x, y)$	cost of (x, y)
T	a tree subgraph of G representing a RST
E_T	set of links in T
G_B	STAR bridge graph of G
E_B	set of edges in G_B
$nca(x, y)$	nearest common ancestor of x and y
$d_T(x, y, G)$	tree path distance between x and y in G
$d_r(x)$	tree path distance between x and the root bridge r

Table 1: Notations

Bridge x is a direct neighbor of bridge y, and vice versa, if $(x, y) \in E$. Let $V = B \cup D$. $T = (V, E_T)$ is a tree subgraph of G representing a RST, wherein $(x, y) \in E_T$ if and only if (x, y) is an activated link in the RST. We refer to the links in E_T as *tree links* and the links in $E - E_T$ as *non-tree links*. If $(x, y) \in E_T$, x and y are *tree neighbors*. A path in T is a *tree path*. A tree path originating at bridge s and terminating at bridge t is denoted $treepath(s, t)$. The length of this tree path is denoted $d_T(s, t)$. Note that $d_T(x, y) = c(x, y)$ if x and y are tree neighbors. The tree path distance between x and the root bridge r is denoted $d_r(x)$. We refer to $treepath(s, t)$ as an *STD bridge tree path* if it has at least one intermediate bridge (i.e., one other than the source and destination bridges) and every intermediate bridge on the path is an STD bridge. If s and t are STAR bridges, and there is an STD bridge tree path between them, then s is a *distant STAR neighbor* of t, and vice versa.

The *nearest common ancestor* of x and y is the highest-level bridge on a tree path between x and y. If x is an ancestor of y, then x is necessarily the nearest common ancestor of x and y. Let the nearest common ancestor of x and y be denoted $nca(x, y)$. We say x and y are on different branches if $nca(x, y) \neq x$ and $nca(x, y) \neq y$. We call $(x, y) \in E - E_T$ a *crosslink* if x and y are on different branches. In this case, x and y are called *crosslink neighbors*. Figure 3 is an example of an undirected graph of a bridged LAN. Node r is the root. The white nodes are STD bridges and black nodes are STAR bridges. The solid lines are tree links and the dotted lines are non-tree links. Link (w, z) is a non-tree link but not a crosslink, while (x, a), (y, u) and (v, q) are all crosslinks. Therefore, w and z are direct neighbors but neither tree neighbors nor crosslink neighbors. They are distant STAR neighbors though since the path $w \rightarrow a \rightarrow z$ is an STD bridge tree path. Table 2 summarizes the definitions of different kinds of neighbors.

Neighbor Type	Formal Definition	Examples
Direct neighbors	$x, y \in V, (x, y) \in E$	r and p, v and q, w and z
Tree neighbors	$x, y \in V, (x, y) \in E_T$	r and p, x and y, w and a
Crosslink neighbors	$x, y \in V, (x, y) \in E - E_T$ and $nca(x, y) \neq x$ and $nca(x, y) \neq y$	x and a, v and q, y and u
Direct STAR neighbors	$x, y \in B, (x, y) \in E$	v and q, x and y, w and z
Distant STAR neighbors	$x, y \in B, treepath(x, y)$ is an STD bridge tree path	w and z, x and v, w and p

Table 2: Neighbor Types

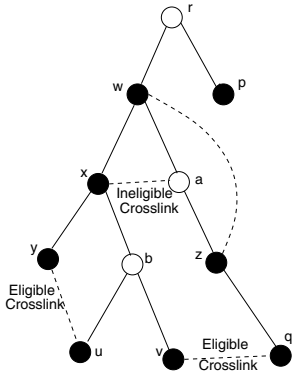


Figure 3: Example of a Bridged LAN Graph

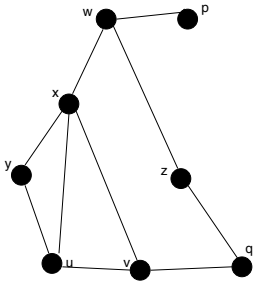


Figure 4: STAR Bridge Graph

Since an STD bridge disables the port associated with a non-tree link, a non-tree link may be used to construct alternate paths only if it connects two STAR bridges. We call a non-tree link that is obviously not used by the STAR protocol, as just described, for supporting any shortest path an *ineligible link*. We call a non-tree link *eligible* otherwise. In Figure 3, (x, a) is an ineligible link while (y, u) and (v, q) are eligible links.

To facilitate the discussion of the STAR path finding process (see next section), which tries to find shorter alternate paths among STAR bridges, we define an overlay graph of a bridged LAN graph. This overlay graph is called the *STAR bridge graph* and it consists of STAR bridges only. We denote the STAR bridge graph of a bridged LAN graph $G = (B, D, E)$ as G_B. There is an edge between two STAR bridges in the STAR bridge graph only when the two STAR bridges are neighbors, either direct or distant, in G. Definition 1 is the formal definition of the STAR bridge graph of a bridged LAN graph. The STAR bridge graph of Figure 3 is shown in Figure 4.

DEFINITION 1. *A STAR bridge graph G_B is defined to be (B, E_B), where $(x, y) \in E_B$ if and only if x and y are STAR neighbors, either direct or distant. $c'(x, y)$, the cost of link (x, y), depends what kind of STAR neighbors that x and y are:*

STAR Neighbor Type	*Value of $c'(x, y)$*
direct but not distant	$c(x, y)$
both direct and distant	$min(d_T(x, y), c(x, y))$
distant but not direct	$d_T(x, y)$

4.3 STAR Path Finding Process

The goal of this process is to compute the Bridge Forwarding (BF) Table. In the best case, the BF Table has next hop and forwarding port information associated with a shortest path to a STAR bridge. The STAR bridge graph contains all tree paths among STAR bridges and all eligible non-tree links in the original bridged LAN. Therefore, the shortest path in G_B between a pair of STAR bridges x and y would be the best path we can achieve in the bridged LAN after pruning ineligible links. Ideally, if we can compute every $c'(x, y)$ correctly, each STAR bridge can compute its own BF Table based on distance vectors.

In a conventional distance vector protocol, each node initializes its distance vector with distances to all its neighbors. It then sends the distance vector to all its neighbors. When a neighbor receives the distance vector, the neighbor updates its own distance vector if any shorter path is found. This neighbor then sends its update to all its own neighbors. The procedure keeps going on until the algorithm converges. Once the algorithm converges, there is no loop in the paths. Interested readers can refer to [19] for the details of the distance vector protocol.

When there are STD and STAR bridges in the bridged LAN, the conventional distance vector update protocol cannot be applied directly. The first issue is to discover neighbors and to find out the distances to all neighbors. In the bridged LAN graph, some neighbors are not direct neighbors but are distant STAR neighbors. For example, x and v in Figure 4 are neighbors but they are not direct neighbors in Figure 3. x and v do not know $d_T(x, v)$ since they only know the cost to a direct neighbor. Moreover, x does not know the existence of v and vice versa. Therefore, as a first step of the STAR path finding process, each STAR bridge has to discover every bridge y that is a distant STAR neighbor and determine $d_T(x, y)$. Unfortunately, due to the limitation of STD bridges, STAR bridges may not be able to determine every distance correctly. For those distances that STAR bridges cannot determine correctly, overestimates are obtained as described in Section 4.3.1.

We call the step of discovering neighbors and estimating the distances to neighbors the *Distance Vector Estimation* procedure. The procedure that follows Distance Vector Estimation is the *Distance Vector Enhancement* procedure. The Distance Vector Enhancement procedure exchanges distance vector information to discover other non-neighbor STAR bridges and to find the shortest paths to them as in the conventional distance vector protocol.

4.3.1 Distance Vector Estimation

Each STAR bridge maintains a distance vector (DV) to keep the information of other STAR bridges. When an unknown STAR bridge is newly discovered, a new entry is created. Each entry in the distance vector consists of a tuple of six fields. Table 3 provides a summary. We denote the entry about the information of STAR bridge k in the distance vector of STAR bridge n as $DVT(n, k)$. The information in $DVT(n, k)$ consists of:

Field	Definition
$d(n,k)$	Estimated distance between n and k
$F(n,k)$	Forwarding port for k
$next(n,k)$	ID of the next hop STAR bridge neighbor on the path from n to k
$FG\_A(n,k)$	Distance accuracy flag with a value `True` if $d(n,k)$ is accurate, and `False` otherwise
$FG\_T(n,k)$	Tree path flag with a value `Tree` if the path from n to k is a tree path, `NonTree` otherwise
$FG\_R(n,k)$	Relation flag with a value `Anc` if k is an ancestor of n, `Dec` if k is a descendant of n, and `Null` otherwise

Table 3: Fields in $DVT(n,k)$ for a path from n to k

(1) an estimated distance between n and k

(2) forwarding port of k

(3) next hop STAR bridge neighbor on the path to k

(4) a flag indicating whether the estimated distance is accurate

(5) a flag indicating whether the path is a tree path

(6) a flag indicating the relation between n and k

As mentioned earlier, this procedure has to discover distant STAR neighbors and estimate their distances. To do this, distant STAR neighbors exchange messages among them. These messages are encapsulated as data frames to bypass intermediate STD bridges between two communicating STAR neighbors. There are two kinds of bridge frames serving for this purpose - `DVMyInfo` and `DV-OurInfo` frames. `DVMyInfo` frames are used for a STAR bridge to inform other STAR bridges of its own topology information. `DVOurInfo` frames carry information related to both the source and the destination STAR bridges. Let's denote the parent of n as $parent(n)$. The content of the `DVMyInfo` frame sent from n, is $< n, d_r(n), parent(n), c(n, parent(n))>$,denoted `DVMyInfo`$(n)$. The content of the `DVOurInfo` frame sent from n to k, denoted `DVOurInfo`(n,k), is $< n, k, d(n,k) >$.

After the RST is built, every bridge k knows its tree links, as well as the root bridge and the root path distance, $d_r(k)$, where bridge r is the root bridge. Each STAR bridge k then sends a `DVMyInfo`(k) frame on its root link if its parent is an STD bridge. This frame is encapsulated as a normal data frame using a multicast address that belongs to the group of STAR bridges as the destination address, the parent STD bridge will forward the frame over all its tree links, except the incoming one.

If there is any STAR bridge along the root path of k, the one that is nearest to k, say n, will receive the `DVMyInfo`(k) frame from a child link (Figure 5(a)). Note that n and k are distant STAR neighbors. Bridge n can determine the tree path distance between k and n by subtracting $d_r(k)$ from $d_r(n)$. That is, $d_T(n,k) = d_T(k,n) = d_r(n) - d_r(k)$. Then, n informs k of the distance between them by a `DVOurInfo`(n,k) frame and stops forwarding the `DVMyInfo`(k) frame. This is the case where n and k are on the same branch and n is an ancestor of k.

If the STAR bridges are on different branches, like v and v' in Figure 5(b), and there is no STAR bridge on the tree path between them, then, v' will receive the `DVMyInfo`(v) frame of v and vice versa. However, there is no way for them to calculate the real tree path distance between them using root path distance alone. In the case of v and v', if they know that they have the same parent, they can determine the accurate tree path distance. Therefore, the `DVMyInfo`(v) frame also contains the information of the parent of v. When v and v' receive each other's `DVMyInfo` frame through a root link and find out that they are siblings, they may calculate the distance between them correctly by adding $c(v, parent(v))$ and $c(v', parent(v'))$. Unfortunately, there are still situations where a pair of STAR bridges cannot determine their tree path distance correctly. Figure 5(c) shows an example of such scenarios. In that example scenario, n can get an overestimate of $d_T(n,j)$ by calculating $d_r(n)+d_r(j)$, or $d_r(n)+d_r(j)-2*d_r(k)$ after discovering k.

After discovering all distant STAR neighbors, each STAR bridge n then fills out $DVT(n,k)$ (Table 3) if k is a direct STAR neighbor of n. If k is a tree neighbor, $d(n,k)$ can be accurately determined to be $c(n,k)$. If k is a distant STAR neighbor, the current $DVT(n,k)$ should contain the information of the tree path between n and k. That is, $d_T(n,k)$ has been obtained, although it may be only an estimate. In order to use the shortest path between n and k, we should assign $d(n,k)$ to be the value $min\{d_T(n,k), c(n,k)\}$. Unfortunately, $d_T(n,k)$ may be incorrect. To avoid selecting a link with a larger distance than its corresponding tree path, we do not replace $DVT(n,k)$ even though $c(n,k) < d_T(n,k)$.

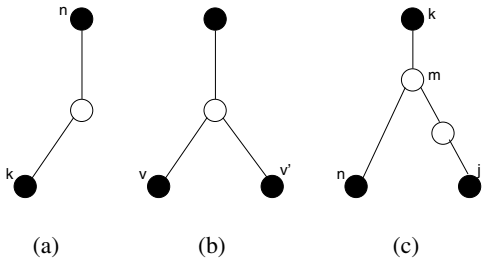


Figure 5: Tree Path Computation Examples

4.3.2 Distance Vector Enhancement

The DV Enhancement procedure is similar to the distance vector exchange procedure in the traditional approach except that we can replace an existing path in the distance vector only when the existing path in the DV is not a tree path, or it is a tree path with its distance accurately obtained. That is, we never replace a tree path which distance is not accurately determined. On the other hand, if a tree path is discovered and we cannot determine its distance correctly, we put this path in the distance vector no matter whether the estimated tree path distance is smaller than the existing path or not. Then, we can ensure that we never use a path which is longer than the tree path. After the DV Estimation procedure, STAR bridge n only knows the tree distance, either correct or

estimate, to its tree STAR neighbors and distant STAR neighbors. In order to let a STAR bridge identify whether a path to a formerly unknown bridge is a tree path and whether the tree path distance is correct, we have to put the accuracy flag and the tree path flag in the DVRecord frames. The content of the DVRecord frame sent from n that contains the information of a path from n to k, denoted DVRecord(n, k), is $< n, k, d(n,k), FG\_A(n,k), FG\_T(n,k), FG\_R(n,k) >$.

When n receives DVRecord(j, k), it discovers a path from n to k that passes through j. This is a newly discovered path from n to k. This newly discovered path is a tree path if the path from n to j and the path from j to k are both tree paths. If this newly discovered path is $treepath(n,k)$ and $d_T(n,k)$ is an estimate, it replaces the existing $DVT(n,k)$. If $d_T(n,k)$ is accurate, it replaces the existing $DVT(n,k)$ when $d_T(n,k)$ is the same or smaller than $d(n,k)$. If the newly discovered path is a non-tree path and the existing $d(n,k)$ is accurate, the existing $DVT(n,k)$ is replaced if the newly discovered non-tree path is shorter. If the existing $DVT(n,k)$ is a tree path and $d_T(n,k)$ is an estimate, no replacement can occur. The following is the pseudocode when n receives DVRecord$(j,k) = < j, k, d, FG_A, FG_T, FG_R >$.

```
if FG_T(n, j) = Tree and FG_T = Tree
    /* the new path is a tree path */

    if FG_A(n, j) = False or FG_T = False
        /* tree path distance is an estimate */
        replace DVT(n, k) by the new tree path
    else
        if d(n, j) + d <= d(n, k)
            replace DVT(n, k) by the new tree path
        endif
    endif

else
    /* the new path is not a tree path */

    if FG_A(n, k) = True or FG_T(n, k) = NonTree
        /* current path in DV can be replaced */

        if d(n, j) + d < d(n, k)
            replace DVT(n, k) by the new path
        endif

    endif

endif
```

Suppose that $d(n,j) = 2$, $FG\_A(n,j)$ = `True`, and $FG\_T(n,j)$ = `Tree`, while $d(n,k) = 6$, $FG\_A(n,k)$ = `True`, and $FG\_T(n,k)$ = `Tree`. Assume that n receives a DVRecord$(j,k) = < j, k,$ 3, `False`, `NonTree`, `Null` $>$. The newly discovered path from n to k is a non-tree path with estimate distance of 2+3 = 5. As the distance of the existing path from n to k, $d(n,k)$ is accurately known ($FG\_A(n,k)$ = `True`), and $5 < d(n,k)$, we can replace $DVT(n,k)$ by the newly discovered path. Although the accurate distance of the newly discovered path is not known, as we obtain overestimates only according the discussion of the previous section, the actual distance is less than 5, which is less the $d_T(n,k)$. That is, the newly discovered path is shorter than the tree path.

Subsequently, when the distance vector becomes stable, $d(n,k) \leq d_T(n,k)$ for all $n, k \in B$ such that $n \neq k$. This stable distance vector is the Bridge Forwarding (BF) Table for forwarding purpose. We refer to a path in the BF Table between a pair of STAR bridges as a *STAR forwarding path*. Note that a STAR forwarding path may be a standard tree path or an enhanced forwarding path when it can be identified.

4.4 STAR Learning Process

The BF Table contains the forwarding information to STAR bridges. However, as the destination address in a data frame is the address of a host, not a STAR bridge, the BF Table alone is not sufficient for frame forwarding. In order to forward a data frame using the BF Table, a STAR bridge has to know which STAR bridge the frame should be sent to. Intuitively, the STAR bridge that is "closest" to the destination host should be the destination STAR bridge. We call this "closest" bridge the *agent bridge* of a host. The STAR learning process is responsible to learn which STAR bridge is the agent bridge to a host, and keeps the information in the Host Location (HL) Table.

4.4.1 Designated Bridge and Agent Bridge

According to the standard, each LAN has a *designated bridge* and this bridge is also the designated bridge of all hosts attached to that LAN. A designated bridge may be an STD bridge or a STAR bridge. The designated bridge is the closest bridge to a host and it is a suitable candidate to be an agent bridge. However, as an agent bridge must be a STAR bridge, when the designated bridge is an STD bridge, another bridge has to be found as the agent bridge. In this case, we select the nearest STAR ancestor of the STD designated bridge to be the agent bridge.

To see why we select agent bridge in this manner, we first observe that the tree path from a host s to a downstream host t is the shortest path already according to the standard spanning tree algorithm. Therefore, an enhanced forwarding path is useful only when s and t are on different branches. The corresponding tree path from s to t in this case goes upstream and then downstream on the tree. To use an enhanced forwarding path, one of the bridges along this tree path must be a STAR bridge so that when the frame passes through that STAR bridge, the STAR bridge can forward it over an enhanced forwarding path. Intuitively, the closer this STAR bridge is to s, the more forwarding path enhancement is achieved. Moreover, this STAR bridge must be on the upstream path from s to t because a downstream tree path is always a shortest path. As a result, we select the nearest STAR ancestor as the agent bridge. Referring to Figure 3, suppose that the designated bridge of host s is STD bridge b. Then the agent bridge of s would be STAR bridge x. There are situations that no agent bridge is identified. For example, if the designated bridge is the root that is an STD bridge, then all STAR bridges will be on downstream and so no bridge will be the agent bridge.

4.4.2 Host Location Table

Each entry in the HL Table of STAR bridge n is a tuple $(s, ab(s))$, where s is a host and $ab(s)$ is the agent bridge of s. Each entry in the Forwarding Database (FD) provided by the standard protocol indicates a forwarding port for a host s. A host s is referred to as a known host with respect to n if either the HL Table or the FD or both have an entry for s. In addition to filling out the HL Table and FD, n should be able to identify whether it is the agent bridge of a host s when a normal data frame from s is received.

When STAR bridge n receives from a child port a data frame that is originated from an unknown host s, it declares itself as the agent bridge of s by broadcasting a HostLoc frame to all other STAR bridges and making a new entry in its HL table. It also makes a new entry in its FD. It sends the HostLoc frame before forwarding the data frame. Therefore, all other STAR bridges will receive

the HostLoc frame of s before the data frame. When a STAR bridge receives the HostLoc frame of s, it updates the location of s. Since n sends the HostLoc frame before the data frame, by the time another STAR bridge receives the data frame, the ancestor STAR bridge has already have the location of s and will not declare itself as the agent bridge of s.

In the standard, the entries in the FD time out periodically so that a host can move to a different LAN without changing its address. The entries in a HL Table time out as the entries in an FD do. Therefore, the size of the HL Table is at most the same size as the FD.

4.5 STAR Forwarding Process

STAR bridges execute the STAR forwarding process after the STAR learning process when a data frame is received. Having received a data frame destined for a host t, a STAR bridge n first checks its HL Table to determine if it knows $ab(t)$, the agent bridge of t. If $ab(t)$ is found, n will then find out from its BF Table the forwarding port of $ab(t)$ and forward the frame accordingly. If n itself is $ab(t)$, it forwards the frame according to the FD. If $ab(t)$ is unknown, n will proceed to check its FD. If host t is unknown, STAR bridge n will forward the data frame on all tree ports except the incoming one, just as the IEEE 802.1D standard.

In the IEEE 802.1D standard, although a bridge may forward the same frame to more than one port, only one port leads to the destination since there is a unique path from any source to any destination on a spanning tree. Therefore, an STD bridge can never receive the same data frame more than once. As the STAR bridge graph may not be a tree, two STAR bridges may receive the same data frame and try to forward it to the destination using different paths. For example, suppose that the designated bridge of the destination host is y and the designated bridge of the source host is b in Figure 3. If b forwarded one copy of the data frame to x and one to u, y would receive two copies of the same data frame, one from x and one from u. The STAR Bridge Protocol avoids this by allowing only x, the agent bridge of s, to forward the frame using, in this case, an enhanced forwarding path. Since the agent bridge is unique, at most one copy of the frame may be sent to the destination.

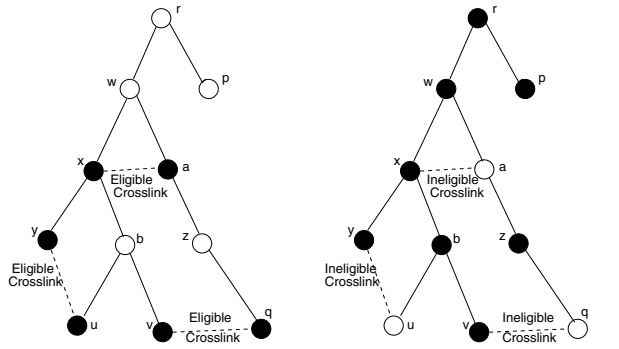
To let an intermediate STAR bridge know whether a frame is intended to be forwarded over an enhanced forwarding path or a tree path, the agent bridge encapsulates the frame if it is sending it over an enhanced forwarding path. The destination address of the encapsulated data frame is the next hop STAR bridge on the enhanced forwarding path. The mechanism in [21] can be used for STAR bridges to identify encapsulated data frames from normal ones. The encapsulation also avoids a frame to be dropped by intermediate STD bridges and redundant traversal of the frame. We refer interested readers to [12] for further details.

The following is the pseudocode of the STAR forwarding process when receiving a data frame sent from host s to host t:

```
if data frame is encapsulated

    if data frame is meant for this bridge

        if this bridge is ab(t)
            forward using FD
        else
            forward using BF Table

    else

        drop the frame

    endif

else

    if this bridge is ab(s)

        if ab(t) is known and is on another branch
            forward using BF Table
        else
            forward using FD
        endif

    else

        forward using FD

    endif

endif
```

5. STD BRIDGE REPLACEMENT STRATEGY

Due to the limitation of STD bridges, routing performance may not be improved significantly when STD bridges are replaced by STAR bridges arbitrarily in an extended LAN. On the other hand, it is not necessary to replace all STD bridges for any performance gain. For example, after placing the six STAR bridges in Figure 6(a), all crosslink links become eligible and can be used to support shorter alternate paths among STAR bridges. On the contrary, in Figure 6(b), even though there are eight STAR bridges, no crosslink becomes eligible and all data frames still have to go through tree paths.



(a) Good Replacement Locations

(b) Bad Replacement Locations

Figure 6: Replacement Examples

We describe in this section a centralized heuristic algorithm that allows an administrator to determine which STD bridges in an extended LAN should be replaced by STAR bridges subject to a limit on the number of STD bridges replaced. We assume that the RST and eligible crosslinks (or possibly added crosslinks) are known before applying the algorithm. The algorithm iteratively solves a local optimization problem to determine a set of STD bridges or a single STD bridge to be replaced by STAR bridges, such that a selected crosslink can be utilized for maximum performance gain. The algorithm stops when further replacement of STD bridges will exceed the limit on the number of STD bridges replaced, or addi-

tional performance gain is not worthwhile with further STD bridge upgrade.

Our problem of bridge replacement is the selection of a subset of STD bridges in an extended LAN so that their upgrade to STAR bridges would result in maximum improvement in forwarding path distances in the extended LAN. In this section, we formulate this optimization problem in terms of an optimization objective and a constraint on the number of STD bridges that may be replaced. This optimization problem is difficult to solve when the size of the problem is large, as the number of subsets of a given size in a large set of bridges is exponentially large. Moreover, if a subset is large, it is very difficult to evaluate the performance gain with respective to the optimization objective. We propose to address the problem iteratively by solving in each iteration a local optimization problem, wherein the extended LAN may have one or some STD bridges, or none, already replaced by STAR bridges, and the number of additional STD bridges that may be upgraded is limited to a predetermined small number. In the local optimization, we identify a small number of STD bridge subsets as *candidate bridge sets* and evaluate the performance gain for each subset in terms of the reduction in the average pair-wise distance of all bridges in G.

The candidate bridge set that offers the maximum performance gain is the *best candidate set*. We then replace the STD bridges that are in the best candidate set by STAR bridges. If further replacement is allowed, we again identify the best candidate set on the existing extended LAN configuration and replace the STD bridges in that set.

5.1 Candidate Bridge Sets

We propose to identify a candidate bridge set with respect to a crosslink, since crosslinks are used to form enhanced forwarding paths. We consider two criteria for defining candidate bridge sets. First, the total number of candidate bridge sets must be polynomially bounded. Second, the calculation of the performance gain of a candidate bridge set must be simple.

We consider a bridge to be *related* to a crosslink if it is necessary for that bridge to be a STAR bridge in order to use that crosslink. Two bridges are considered *related* bridges if they are both related to the same crosslink. In order to use a crosslink (x, y), bridges x and y have to be STAR bridges, and the tree path distance between them has to be correctly calculated by the STAR bridge protocol. This requires the nearest common ancestor of bridges x and y, $nca(x, y)$, to be a STAR bridge too. Hence, bridges x, y, and $nca(x, y)$ are related bridges and they all are related to crosslink (x, y). In this case, there is a candidate bridge set specified by $R_{(x,y)} = \{x, y, nca(x, y)\}$.

In general, at most three bridges may be related to a crosslink. When there are no STAR bridges in an extended LAN, activating a crosslink requires replacing all the STD bridges related to the crosslink. After some STD bridges have been replaced, it may not be necessary to replace all STD bridges related to a crosslink to activate it since one or more bridges that are related to the crosslink may have already been upgraded to be STAR bridges. Denoting a candidate bridge set associated with an inactive crosslink (x, y) by $K_{(x,y)}$, we have $K_{(x,y)} = R_{(x,y)} \cap D$, where D represents the set of STD bridges. We refer to such a candidate bridge set as a *crosslink activation* candidate bridge set.

When there are one or more activated crosslinks in an extended LAN, we may not need to activate new crosslinks in order to support a new enhanced forwarding path. For example, suppose that crosslink (x, y) with cost $c(x, y)$ has been activated and p is an STD bridge on the tree path from $nca(x, y)$ to y as shown in Figure 7. Let's further assume that $d_T(p, y) + c(x, y) < d_T(p, x)$. Then, if we replace p by a STAR bridge, the path between p and x will be enhanced. Such a candidate bridge set, which contains a single STD bridge p, is referred to as a *path enhancement* candidate bridge set and simply denoted K, wherein $K = \{p\}$. Note that a path enhancement candidate bridge set may be a subset of a crosslink activation set. For example, the path enhancement candidate bridge set $\{p\}$ is a proper subset of the crosslink activation candidate bridge set $K_{(p,q)} = \{p, q\}$. In this case, the performance gain of $\{p\}$ must be smaller than the performance gain of $\{p, q\}$ and we can exclude it from replacement consideration.

In summary, a candidate bridge set is either a set that contains only one STD bridge, or two or three related STD bridges. Note that the number of candidate bridge sets is a bridged LAN graph $G = (B, D, E)$ is at most $|D|$ + *number of inactive crosslinks* in E, where B is the set of STAR bridges, D is the set of STD bridges, and E is the set of links connecting the bridges. As the number of inactive crosslinks is at most $|E|$, the total number of candidate bridge sets is $O(|E|)$.

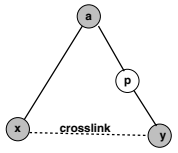


Figure 7: Example

5.2 Agent Bridge for STD Bridges

An enhanced forwarding path from host s to host t must traverse the agent bridge of s and the agent bridge of t. As illustrated in the proof of Theorem 1, the enhancement of the path from $db(s)$ to $db(t)$ depends on the enhancement of the path from $ab(s)$ to $ab(t)$. In this section, we define the agent bridge of an STD bridge for the evaluation of performance gain.

Let $ab(a, G)$ denote the agent bridge of an STD bridge a in G. $ab(a, G)$ is defined to be the STAR bridge that is the agent bridge of any end station whose designated bridge is the STD bridge a. We denote the designated bridge of a host s in G as $db(s, G)$. Note that the designated bridge of a host does not change no matter where STAR bridges are located. Then, $ab(a, G) = ab(s, G)$ if s is a host and $db(s, G) = a$. In other words, the agent bridge of STD bridge a is the nearest ancestor STAR bridge of a. If a does not have a STAR ancestor, $ab(a, G)$ is undefined and denoted as $ab(a, G) = \phi_{ab}$. For simplicity, we define $B' = B \cup \{\phi_{ab}\}$, and the agent bridge of a STAR bridge x in G to be itself, i.e., $ab(x, G) = x$.

Define $A(x, G)$ to be the set of bridges whose agent bridge is x, i.e., $A(x, G) = \{a | ab(a, G) = x, a \in G, x \in B'\}$. By definition, $x \in A(x, G)$ for every $x \in B$, and $A(\phi_{ab}, G)$ is the set of bridges with undefined agent bridge.

Consider the bridged LAN graph in Figure 8, where dark nodes are STAR bridges and white nodes are STD bridges. In this example, we have $ab(r, G) = ab(b, G) = ab(d, G) = \phi_{ab}$. $A(\phi_{ab}, G) = \{b, d, r\}$, $A(x, G) = \{a, f, x\}$, $A(y, G) = \{j, y\}$, and $A(z, G) = \{g, h, z\}$.

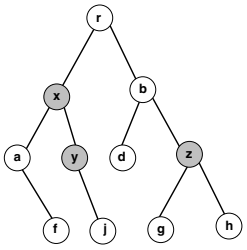
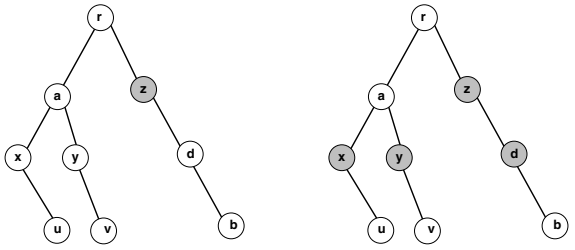


Figure 8: Agent Bridge Example



(a) G (b) G^K

Figure 9: Lemma Example

Let G^K represent a bridged LAN graph after replacing STD bridges in K by STAR bridges. Formally, $G^K = (B \cup K, D - K, E)$. The following lemmas capture a few properties of agent bridges. These lemmas are provided without proof, as they can be verified in a straight forward manner.

LEMMA 1. *If the agent bridge of a is not defined after upgrading STD bridges in K to be STAR bridges, then the agent bridge of a is also not defined before the upgrades. Formally, $ab(a, G^K) = \phi_{ab} \Rightarrow ab(a, G) = \phi_{ab}$.*

When $ab(a, G^K)$ is undefined, it implies that a must be a STD bridge. This lemma also implies that upgrading STD bridges in K to be STAR bridges cannot remove any existing agent bridge ($ab(a, G) \neq \phi_{ab} \Rightarrow ab(a, G^K) \neq \phi_{ab}$). An existing agent bridge of an STD bridge a in $D - K$ may however be substituted by one of the bridges in K to be the new agent bridge of a.

LEMMA 2. *If the agent bridge of b is a bridge d after the upgrades, then the agent bridge of b and the agent bridge of d before the upgrades are the same. Formally, $ab(b, G^K) = d \neq \phi_{ab} \Rightarrow ab(b, G) = ab(d, G)$.*

Suppose that upgrading STD bridges in K to be STAR bridges activates one of them, d, as illustrated in Figure 9 to become an agent bridge of an STD bridge b in $D - K$. If b has an agent bridge $ab(b, G)$ to begin with, then the new agent bridge d must be located along the tree path between b and its original agent bridge $ab(b, G)$ (z in Figure 9). Otherwise, the original agent bridge of b cannot be substituted by the new agent bridge. If b has no agent bridge to begin with, then its new agent bridge d will have no agent bridge itself in the original bridged LAN graph because there is no STAR bridge upstream of d in the original bridged LAN graph G.

LEMMA 3. *Let the agent bridges of u and v after the upgrades be x and y respectively. If x and y are on different branches, then the nearest common ancestor of u and v is the same as the nearest common ancestor of x and y.*

The agent bridge of u is x implies that u is on the downstream of x. Similarly, v is on the downstream of y. As x and y are on different branches, the tree path from u to v must go through x and y as shown in Figure 9. It follows that $nca(u, v) = nca(x, y)$.

5.3 Performance Gain

We define the performance gain due to a candidate bridge set K in a bridged LAN graph $G = (B, D, E)$ in terms of the difference between the average pair-wise distances of bridges in G with and without replacement of bridges in K by STAR bridges.

Given $d(x, y, G)$, the length of the STAR forwarding path between bridges x and y for each pair of bridges x and y in G, the average pair-wise distance in G is

$$d_{avg}(G) = \frac{1}{|V| * (|V| - 1)} \sum_{x,y \in V, x \neq y} d(x, y, G) \qquad (1)$$

where $V = B \cup D$.

$P(K)$, the performance gain of a candidate bridge set K, is defined as follows.

$$P(K) = \{|V| * (|V| - 1)\} * \{d_{avg}(G) - d_{avg}(G^K)\} \qquad (2)$$

A STAR forwarding path from a bridge x to another bridge y is the forwarding path for a data frame that is sent from an end station s whose designated bridge is x to an end station t whose designated bridge is y. We assume that $d(x, y, G) = d(y, x, G)$.

If both x and y are STAR bridges, then $d(x, y, G) = d(x, y, G_B)$, where G_B represents the STAR bridge graph of G.

LEMMA 4. *If bridges x and y are on the same branch, then upgrading bridges in K to be STAR bridges offers no improvement to the length of the forwarding path between x and y. Furthermore, $d(x, y, G) = d_T(x, y, G) = d_T(x, y, G^K) = d(x, y, G^K)$.*

Lemma 4 comes from the fact that a tree path between two bridges on the same branch is always a shortest forwarding path. If bridges x and y are on different branches, then $d(x, y, G)$ depends on whether the agent bridges of x and y exist as well as the relationship between the agent bridges. If both agent bridges exist and are on the same branch, the tree path is the shortest path that is available between them and there is no enhanced forwarding path. If both agent bridges exist and are on different branches, then the STAR forwarding path consists of a tree path from x to $ab(x, G)$, a STAR forwarding path from $ab(x, G)$ to $ab(y, G)$, and a tree path from $ab(y, G)$ to y. Otherwise, the STAR forwarding path is simply a tree path.

In general, if bridges x and y are on different branches, we have

$d(x, y, G) =$

$$\begin{cases} d_T(x, ab(x,G), G) & ab(x,G) \neq \phi_{ab} \text{ and} \\ +d(ab(x,G), ab(y,G), G) & ab(y,G) \neq \phi_{ab} \text{ and} \\ +d_T(ab(y,G), y, G), & FG\_R(ab(x,G), ab(y,G)) = \\ & \texttt{Null} \\ \\ d_T(x, nca(x,y), G) & \\ +d_T(nca(x,y), y, G) & \text{otherwise.} \end{cases} \quad (3)$$

where the value of the relation flag FG_R being `Null` means that $ab(x, G)$ and $ab(y, G)$ are on different branches.

LEMMA 5. *If bridges x and y are on different branches, and at least one of their agent bridges is not defined in G^K, then $d(x, y, G) = d(x, y, G^K)$.*

Since at least one of the agent bridges of x and y is not defined in G^K, Lemma 1 implies that the corresponding agent bridge is not defined in G either. In this case, we have $d(x, y, G) = d_T(x, nca(x, y), G) + d_T(nca(x, y), y, G) = d_T(x, nca(x, y), G^K) + d_T(nca(x, y), y, G^K) = d(x, y, G^K)$, wherein the first step is due to (3) and the second step is due to Lemma 4.

LEMMA 6. *Suppose that bridges u and v are on different branches of the spanning tree. If $ab(u, G^K) = x$, $ab(v, G^K) = y$, such that x and y are different STAR bridges, then $d(u, v, G) - d(u, v, G^K) = d(x, y, G) - d(x, y, G^K)$.*

PROOF: By Lemma 2, $ab(u, G) = ab(x, G)$ and $ab(v, G) = ab(y, G)$. In addition, $d(u, v, G^K) = d_T(u, x, G^K) + d(x, y, G^K) + d(y, v, G^K)$. Therefore, $d(u, v, G^K) - d(x, y, G^K) = d_T(u, x, G^K) + d_T(y, v, G^K)$ since u is on the downstream of x and v is on the downstream of y.

Case I: $ab(x, G) = \alpha \neq \phi_{ab}$ and $ab(y, G) = \beta \neq \phi_{ab}$ and $FG\_R(\alpha, \beta) =$ `Null`
By (3), $d(x, y, G) = d_T(x, \alpha, G) + d(\alpha, \beta, G) + d_T(\beta, y, G)$. Similarly, $d(u, v, G) = d_T(u, ab(u, G), G) + d(ab(u, G), ab(v, G), G) + d_T(ab(v, G), v, G) = d_T(u, \alpha, G) + d(\alpha, \beta, G) + d_T(\beta, v, G)$. Then, $d(u, v, G) - d(x, y, G) = d_T(u, \alpha, G) - d_T(x, \alpha, G) + d_T(v, \alpha, G) - d_T(y, \alpha, G) = d_T(u, x, G) + d_T(y, v, G)$. As the tree path distances are the same before and after upgrades, we can conclude that $d(u, v, G) - d(x, y, G) = d(u, v, G^K) - d(x, y, G^K)$, implying $d(u, v, G) - d(u, v, G^K) = d(x, y, G) - d(x, y, G^K)$.

Case II: when Case I does not hold
By Lemma 3, $nca(u, v) = nca(x, y)$, say α. By (3), $d(x, y, G) = d_T(x, \alpha, G) + d_T(\alpha, y, G)$. Then, $d(u, v, G) - d(x, y, G) = d_T(u, \alpha, G) - d_T(x, \alpha, G) + d_T(\alpha, v, G) - d_T(\alpha, y, G) = d_T(u, x, G) + d_T(y, v, G) = d(u, v, G^K) - d(x, y, G^K)$. □

We now derive a formula for $P(K)$. In general, $d(u, v, G) - d(u, v, G^K) = 0$ if u and v are on the same branch of the spanning tree. By Lemma 5, the same is true if u and v are on different branches but at least one of their agent bridges is not defined, or their agent bridges are on the same branch in G^K. For all other cases, Lemma 6 applies. Therefore, each non-zero term in $P(K)$, corresponding to the improvement of the length of a forwarding path from a bridge u to another bridge v, must be due to an improvement in the length of a forwarding path from $ab(u, G^K)$ to $ab(v, G^K)$. It follows that $P(K) =$

$$\sum_{x,y \in B \cup K} |A(x, G^K)| * |A(y, G^K)| * (d(x, y, G) - d(x, y, G^K)). \quad (4)$$

5.4 Algorithms

In order to evaluate $P(K)$, we need to find $|A(x, G)|$ and $d(x, y, G) - d(x, y, G^K)$. We now describe how to find $|A(x, G)|$ and $d(x, y, G) - d(x, y, G^K)$ in an efficient way and then present the complete bridge replacement strategy pseudocode.

5.4.1 Size of Agent Bridge Set

$|A(x, G)|$ is at most the size of the subtree rooted at x. If a descendant of x is a STAR bridge, x cannot be the agent bridge of the bridges which are on the subtree rooted at that descendant. Refer to Figure 8, j is on the subtree rooted at x but $j \notin A(x, G)$ even if $x \in B$. Generally speaking, $|A(x, G)| = 1 + |\{ y \mid y \notin B, y$ is a descendant and $d_T(x, y)$ is an STD bridge path$\}|$. All agent bridge set sizes can be found by a postorder traversal of the tree started at the root. That is, in order to find $|A(x, G)|$, $|A(y, G)|$, and $|A(z, G)|$ in Figure 8, we need to traverse the tree once starting at r. The time needed for the traversal is $O(|V|)$.

5.4.2 Distance Between Bridges

Suppose that we know $d(x, y, G)$ before any replacement, finding $d(x, y, G^K)$ will be sufficient to calculate $d(x, y, G) - d(x, y, G^K)$. A straight forward approach to find $d(x, y, G^K)$ is to run all-pair shortest path algorithm on G_B^K. If this is too expensive, a more efficient way to find $d(x, y, G^K)$ is by identifying whether the STAR forwarding path from x to y in G^K traverses any newly added crosslink or newly replaced bridge. Note that if $d(x, y, G_B^K) < d(x, y, G)$, the STAR forwarding path from x to y on G_B^K must pass through a bridge in K. Therefore, $d(x, y, G_B^K) = min\{d(x, y, G), min\{d(x, p, G_B^K) + d(p, y, G_B^K) | p \in K\}\}$.

5.4.3 Complete Algorithm

We now present the complete bridge replacement algorithm. The algorithm is locally optimal with respect to the selection of a single candidate bridge set. It is useful when the number of available STAR bridges is significantly smaller than the total number of bridges in the extended LAN. In each iteration of the algorithm, the candidate bridge set that can achieve maximum performance gain is identified and replaced. The algorithm terminates when there is no more STAR bridge to be replaced due to insufficiency of STAR bridge available or the performance gain is not worthwhile. In each iteration, the following steps are done:

1. identify proper candidate bridge sets
2. for each proper candidate bridge set, K
   (a) find out activated crosslink(s) if any
   (b) construct G_B^K
   (c) find out $|A(x, G^K)|$ for every STAR bridge x in $B \cup K$
   (d) find out $d(x, y, G_B^K)$ for every STAR bridge pair x and y
   (e) find out $P(K)$
3. find the best candidate set
4. update G, and G_B

5. update $d(x, y, G)$ for any $x, y \in V$

Step 1 finds out all proper candidate bridge sets in the bridged LAN graph. A proper candidate bridge set is a candidate bridge set that satisfies two requirements: (1) the size of the set must be at most the number of available STAR bridges; (2) it is not a proper subset of another proper candidate bridge set. As we assume that there is a limit on the total number of STAR bridges available, requirement (1) is used to prune infeasible candidate bridge sets that require more STAR bridges than that are available. Requirement (2) prunes the candidate bridge sets that can never be better than another proper candidate bridge set. We discussed that a candidate can be a proper subset of other candidate in Section 5.1. To find all proper candidate bridge sets, we first identify all proper crosslink activation candidate bridge sets and then proper path enhancement candidate bridge sets. When a proper crosslink activation candidate bridge set is identified, we mark the bridges in that set. Then, after all proper crosslink activation candidates are identified, a path enhancement candidate $\{p\}$ is proper only when p is not marked. This step takes $O(|E|)$ time since there are $O(|E|)$ proper candidate bridge sets.

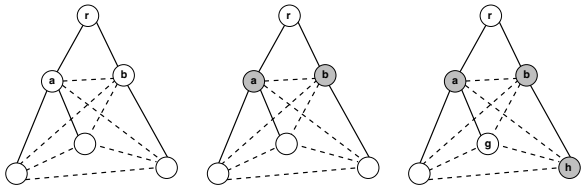
We discussed the steps in Step 2 in earlier sections and the most expensive one is Step 2d. It involves running the Dijkstra's algorithm on G_B^K for every bridge in K and updating $d(x, y, G_B^K)$. Number of nodes in G_B^K is $|B| + |K|$. As $|K| \leq 3$, the time needed to run the Dijkstra's algorithm for every bridge in K is $O(|B|^2)$. Updating $d(x, y, G_B^K)$ takes the same complexity. Therefore, the total time for Step 2 is $O(|E||B|^2)$.

The execution times for Steps 3 and 4 are small compared with other steps. Step 5 takes $O(|V|^2)$. As a result, the total time for each iteration is $O(|E||B|^2)$.

5.4.4 Example

We now use an example to illustrate the whole algorithm. Suppose that Figure 10(a) is the initial G, where all bridges are STD bridges. Assume that the distance function is the hop-count. Solid edges are tree links and dashed edges are crosslinks. There are seven possible crosslinks in the example network as shown in the figure. Suppose that we can replace at most three bridges.

Table 4 shows the distances among all 6 bridges before the replacement algorithm starts, after the first iteration, and after the second iteration.



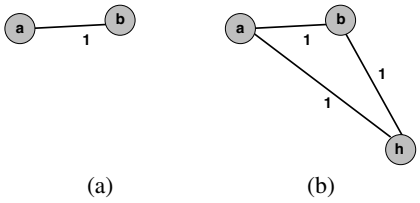
(a) original G (b) after one iteration (c) after two iterations

Figure 10: G

First Iteration

As there is no STAR bridge in G, there is no path enhancement candidate. There are seven crosslink activation candidates, one for each crosslink. They are $K_{(a,b)} = \{a, b\}$, $K_{(a,h)} = \{a, h, r\}$, $K_{(b,f)} = \{b, f, r\}$, $K_{(b,g)} = \{b, g, r\}$, $K_{(f,g)} = \{f, g\}$, $K_{(f,h)} = \{f, h, r\}$, and $K_{(g,h)} = \{g, h, r\}$. $r \notin K_{(a,b)}$ because a and b are siblings and according to the STAR bridge protocol, the tree path distance between a and b can be correctly evaluated even if r is an STD bridge.

As there are no STAR bridges in G, the only STAR bridges in $G^{K_{(a,b)}}$ are those bridges in $K_{(a,b)}$. Therefore, $P(K_{(a,b)})$ = 2 * $|A(a, G^{K_{(a,b)}})|$ * $|A(b, G^{K_{(a,b)}})|$ * $(d_T(a, b, G) - c(a, b))$ = 2 * 3 * 2 * 1 = 12. Similarly, $P(K_{(a,h)})$ = 12, $P(K_{(b,f)}) = P(K_{(b,g)})$ = 8, $P(K_{(f,g)})$ = 2, and $P(K_{(f,h)}) = P(K_{(g,h)})$ = 6. $K_{(a,b)}$ is the best candidate set since it has the maximum performance gain and it requires fewer STAR bridges than $K_{(a,h)}$. Figure 10(b) shows the network after replacing a, b to be STAR bridges. This is the bridged LAN graph which the next iteration is working on. We then update $d(a, b, G^{K_{(a,b)}})$ and some other distances as shown in Table 4, where distances shortened are highlighted in boxes. The STAR bridge graph of Figure 10(b) is in Figure 11(a). This concludes the first iteration.



(a) (b)

Figure 11: G_B^K

	a	b	f	g	h
r	1/1/1	1/1/1	2/2/2	2/2/2	2/2/2
a	-	2/ 1 /1	1/1/1	1/1/1	3/ 2 / 1
b	-	-	3/ 2 /2	3/ 2 /2	1/1/1
f	-	-	-	2/2/2	4/ 3 / 2
g	-	-	-	-	4/ 3 / 2

Table 4: Distances (Initial/After 1st Iteration/After 2nd Iteration)

Second Iteration

After replacing the bridges a and b, we have six crosslink activation candidates. They are $K_{(a,h)} = \{h\}$, $K_{(b,f)} = \{f\}$, $K_{(b,g)} = \{g\}$, $K_{(f,g)} = \{f, g\}$, $K_{(f,h)} = \{f, h\}$, and $K_{(g,h)} = \{g, h\}$. $a \notin K_{(a,h)}$ because a is already a STAR bridge. $r \notin K_{(a,h)}$ because after a becomes a STAR bridge, $d_T(a, h)$ can be correctly determined even r is not a STAR bridge. As we have only 1 STAR bridge left, $K_{(f,g)}$, $K_{(f,h)}$, and $K_{(g,h)}$ are not proper candidate bridge sets. As $d_T(a, h, G)$ is now 2, the performance gain of $\{h\}$ is 2 * $|A(a, G^{K_{(a,h)}})|$ * $|A(h, G^{K_{(a,h)}})|$ * $(d_T(a, h, G) - c(a, h))$ = 2 * 3 * 1 * 1 = 6. The performance gains of $K_{(b,f)}$ and $K_{(b,g)}$ are both 4. $\{h\}$ is the best candidate set and $G_B^{\{h\}}$ is shown in Figure 11(b). Figure 10(c) is the bridged LAN graph after replacing bridge h and Table 4 shows the updated distances between bridges.

The performance gain in terms of hops of the three bridges is 18, which is the sum of the performance gains in the two iterations.

Name	Content	Space Needed
BF Table	Best path between STAR bridges	$O(\|B\|)$
HL Table	Location of hosts	$O(\|M\|)$
FD	Forwarding port of hosts	$O(\|M\|)$

Table 5: Storage Requirements in STAR Bridges

The average normalized reduction in forwarding path, which is avg $\frac{d_T(x,y)-d(x,y)}{d_T(x,y)}$, is 29%. If all seven crosslinks were activated, the total distance shortened would be 28. Let $short(x, y)$ be the shortest distance between x and y. avg $\frac{d_T(x,y)-short(x,y)}{d_T(x,y)}$ = 48%. Therefore, after replacing half of the STD bridges and putting 2 out of 7 crosslinks, the performance is increased by over half of the possible gain. Note also that if only 2 crosslinks are allowed to be added in the network, adding (a, b) and (a, h) yields the maximum enhancement among all other combinations of crosslinks. The STAR bridge protocol is able to identify all possible shortest paths when only (a, b) and (a, h) are in the network.

6. PERFORMANCE OF STAR BRIDGE PROTOCOL

In this section, we analyze the storage, message complexity, and path length of the STAR Bridge Protocol.

6.1 Storage

Each STD bridge keeps only one table for forwarding, which is the FD. One entry is necessary for each known host. Therefore, the space required is $O(|M|)$, where M is the set of all hosts in the extended LAN. In addition to an FD, there are two new tables in each STAR bridge: BF Table and HL Table. Table 5 is a summary of all tables in the STAR bridge n.

After the STAR learning process has been executed for some time and old entries in the FD have been timed out, a host s appears in both the HL Table and the FD of STAR bridge n only if n is the agent bridge of s. Therefore, the total memory needed for the HL Table and the FD in STAR bridges together would be about the same as in the STD bridges. We do need extra space for the BF Table. However, as the number of entries of the BF table is at most $|B|$ which is far less than $|M|$, we can conclude that the storage requirement in a STAR bridge is comparable to that in an STD bridge.

6.2 Message Complexity

For each message generated by the path finding process, there is at most one recipient on each port. The number of messages needed for each pair of STAR bridges depends on the length of the enhanced forwarding path between them. The path length is bounded by the diameter of the tree. The number of messages generated by the spanning tree is related to the diameter of the tree too. Therefore, we can conclude that the number of messages generated by the path finding process is at most $|B|$ times of the messages needed in building the spanning tree.

We have conducted simulation studies to determine empirically the number of messages generated by the path finding process. We generated networks of different sizes of different branching factors. The simulation setup is summarized in Table 6. For each network size with a certain range of root branching factor and a certain range of number of children in other nodes, 50 different topologies were generated. There was a total of 2200 different topologies studied. In each topology, the rooted spanning tree was generated first. The number of children of the root node was either within the range of [4, 6] or [6, 8]. The exact number of children was selected arbitrarily among the numbers in the range. The numbers of children of other nodes are either 0 or in the range specified. The tree link costs were integers in the range [1, 3]. After the tree was generated, 20% - 50% of the nodes were selected to be STAR bridges. The STAR bridges were selected according to the strategy described in Section 5. We assume a crosslink is available between any bridges that are on different branches. The cost of each crosslink must be set in a way that it should not change the tree structure. Therefore, each crosslink was associated with a minimum cost and the minimum cost of crosslink (x, y) is $|d_r(x) - d_r(y)|$. In our simulation, the cost of crosslink was set to be the sum of the minimum cost and an integer in the range [1, 2].

Item	Setting
Network Size	20 - 30 nodes
Number of Children	root node: [4, 6], [6, 8] other nodes: [2, 4], [4, 6]
# of Networks per Setting	50
Link Cost	tree links: [1, 3] other links: min. cost + [1, 2]
Number of STAR Bridges	20% - 50%

Table 6: Simulation Setup

Table 7 shows the message overhead of the STAR path finding process in networks of different sizes. In the table, `Num_Mesg(PF)` is the number of messages generated by the STAR path finding process and `Num_Mesg(ST)` is the number of messages generated to build the rooted spanning tree in the standard protocol. The table shows that `Num_Mesg(PF)` $\leq |B|$ * `Num_Mesg(ST)`. More specifically, `Num_Mesg(PF)` is approximately $\frac{1}{3} * |B| *$ `Num_Mesg(ST)`.

Size	`Num_Mesg(PF)`	`Num_Mesg(ST)`	$\frac{\texttt{Num\_Mesg(PF)}}{\|B\|*\texttt{Num\_Mesg(ST)}}$
20	177	73	0.3603
21	195	78	0.3432
22	191	81	0.3331
23	232	86	0.3455
24	252	91	0.3330
25	310	99	0.3299
26	325	102	0.3384
27	333	106	0.3316
28	366	111	0.3292
29	359	115	0.3147
30	446	125	0.3258

Table 7: Message Overhead of the Path Finding Process

The path finding process will not generate any message after building the BF Table. Nevertheless, the root bridge will keep on generating configuration messages periodically after the spanning tree has been built to maintain the tree. Therefore, for a stable bridged LAN, the extra number of messages generated by the path finding process is negligible.

Location information is necessary in all algorithms described in Section 3 that are applicable for any additive metric. In those algo-

rithms, every bridge has to know the location of all known hosts. In the STAR Bridge Protocol, STAR bridge keeps only the location of host s provided $ab(s)$ is defined. Therefore, the location messages generated by the STAR Bridge Protocol are less than those generated by the algorithms in Section 3.

6.3 Path Length

In this section, we prove that the length of a STAR forwarding path is always less than or equal to the corresponding tree path, and present our simulation results. In the following discussion, we denote the length of the STAR forwarding path between two bridges x and y as $d(x, y)$.

THEOREM 1. *The length of the STAR forwarding path for a frame sent by a host s to another host t is less than or equal to the length of the corresponding IEEE 802.1D tree path. In other words,*

$$d(db(s), db(t)) \leq d_T(db(s), db(t)).$$

PROOF: We first observe that any STAR forwarding path is either a tree path or an enhanced forwarding path. If it is a tree path, the proof is complete. Otherwise, we will show that the inequality still holds.

We divide all situations into the following two cases:

1. Either $ab(s)$ or $ab(t)$ or both are not defined
2. Both $ab(s)$ and $ab(t)$ are defined
   (a) $ab(s)$ and $ab(t)$ are on different branches
   (b) $ab(s)$ and $ab(t)$ are on the same branch

CASE 1: If $ab(s)$ is not defined, no STAR bridge will encapsulate a data frame originated from s. Therefore, the data frame will follow a tree path. If $ab(t)$ is not defined, no enhanced forwarding can be used since there is no entry for t in the HL table.

CASE 2(a): Figure 12 shows various exemplary scenarios for this case. In the figure, a black node represents a STAR bridge, a white node represents an STD bridge, and a dot-dash line represents a tree path. Since $ab(s)$ and $ab(t)$ are on different branches, s and t must be on different branches too. If the forwarding path from $db(s)$ to $db(t)$ is a tree path, then the proof is complete. Otherwise, the forwarding path is an enhanced forwarding path. $ab(s)$ is the first STAR bridge on the enhanced forwarding path and $ab(t)$ is the last STAR bridge on the enhanced forwarding path. Therefore, the enhanced forwarding path consists of three disjoint segments. The first segment, which is a tree path from $db(s)$ to $ab(s)$, has a path length $d_T(db(s), ab(s))$. The second segment, which is an enhanced forwarding path from $ab(s)$ to $ab(t)$, has a path length $d(ab(s), ab(t))$. The third segment, which is a tree path from $ab(t)$ to $db(t)$, has a path length $d_T(ab(t), db(t))$. Therefore, $d(db(s), db(t))$, the length of the STAR forwarding path from $db(s)$ to $db(t)$, satisfies the following inequality.

$$\begin{aligned} & d(db(s), db(t)) \\ = \;& d_T(db(s), ab(s)) + d(ab(s), ab(t)) + d_T(ab(t), db(t)) \\ \leq \;& d_T(db(s), ab(s)) + d_T(ab(s), ab(t)) + d_T(ab(t), db(t)) \\ = \;& d_T(db(s), db(t)) \end{aligned}$$

CASE 2(b): When $ab(s)$ and $ab(t)$ are on the same branch, the tree path is the shortest path between them and the frame will be forwarded using the tree path. □

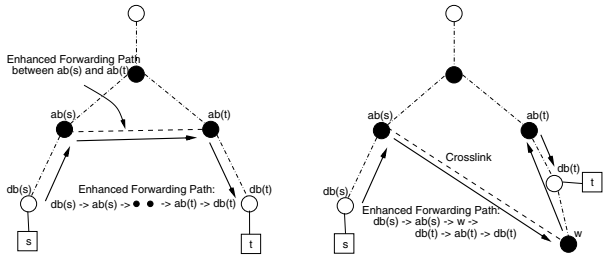


Figure 12: Scenarios for Theorem 1

We now present our simulation results. We compare the performance of the STAR Bridge Protocol with the standard protocol by measuring the maximum path length ratio and the average normalized reduction in forwarding path length. The simulation setting is described in Section 6.2. We denote the shortest path between x and y as $short(x, y)$. The shortest path refers to the shortest path that can be achieved in the network with the activated crosslinks selected by the STD bridge replacement strategy. The maximum path length ratio of the standard is max $\frac{d_T(x,y)}{short(x,y)}$. Similarly, the maximum path length ratio of STAR is max $\frac{d(x,y)}{short(x,y)}$. The average normalized reduction in forwarding path length of STAR bridge protocol $R(STAR)$ is avg $\frac{d_T(x,y)-d(x,y)}{d_T(x,y)}$. As $d_T(x, y) \geq d(x, y)$, a larger value implies more enhancement. An upper-bound on the average normalized reduction in forwarding path length, which is that achieved by shortest paths, is avg $\frac{d_T(x,y)-short(x,y)}{d_T(x,y)}$. Table 8 shows the simulation result. The result shows that the STAR Bridge Protocol outperforms the standard protocol is all network sizes and the average normalized reduction in forwarding path length is about 20%. The reduction is over 95% of that the shortest paths can achieve, demonstrating that the STD bridge replacement strategy effectively identifies appropriate locations for putting STAR bridges. On the other hand, the STAR bridge protocol significantly reduces the maximum path ratios in all networks.

7. CONCLUSION

In this paper, we have described a novel bridge protocol, called STAR, which has provably enhanced forwarding path performance and can be used to improve the QoS routing capability in an extended LAN in a cost-effective manner. As we aim to strike a balance between performance and protocol simplicity, we have chosen to use best-effort shortest paths rather than shortest paths. Specifically, our protocol uses a spanning tree for default frame forwarding, and allows shorter alternate paths to be used whenever they are available and can be identified. Moreover, we require our protocol to be backward compatible with the standard IEEE 802.1D spanning tree bridge protocol to offer smooth migration. Therefore, improvement may be achieved by replacing an appropriate subset of the bridges in an extended LAN by STAR bridges. We have also described a heuristic algorithm for determining appropriate locations for incremental replacement of STD bridges by STAR bridges such that, given a constraint on the number of STD bridges that may be upgraded, the reduction in the average pair-wise distance among bridges in the extended LAN is maximized. Our study shows that we can significantly improve the end-to-end performance when deploying the STAR Bridge Protocol.

Size	$R(short)$ = avg $\frac{d_T(x,y)-short(x,y)}{d_T(x,y)}$	$R(STAR)$ = avg $\frac{d_T(x,y)-d_(x,y)}{d_T(x,y)}$	$\frac{R(STAR)}{R(short)}$	max $\frac{d_T(x,y)}{short(x,y)}$	max $\frac{d(x,y)}{short(x,y)}$
20	0.1997	0.1921	0.9617	7.25	1.0093
21	0.2113	0.2027	0.9594	7.46	1.0108
22	0.2018	0.1931	0.9565	7.44	1.0105
23	0.2042	0.1947	0.9538	7.86	1.0113
24	0.2032	0.1953	0.9612	7.66	1.0095
25	0.2242	0.2169	0.9673	8.15	1.0093
26	0.2152	0.2075	0.9643	8.20	1.0096
27	0.2219	0.2125	0.9578	8.73	1.0117
28	0.2202	0.2138	0.9709	8.13	1.0079
29	0.2092	0.2013	0.9618	8.36	1.0096
30	0.2295	0.2208	0.9621	9.07	1.0111

Table 8: Path Performance

8. ACKNOWLEDGMENTS

The authors would like to thank Art Harvey and the anonymous reviewers for their constructive comments and suggestions on improving the quality of the paper.

9. REFERENCES

[1] *Data-Over-Cable Service Interface Specifications (DOCSIS)*.

[2] *Information technology - telecommunications and information exchange between systems - local an d metropolitan area networks - common specifications. Part 5: Remote Media Access Control (MAC) bridging*, 1998.

[3] *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.

[4] E. Benhamou. Integrating Bridges and Routers in a Large Internetwork. *IEEE Network Magazine*, 2(1), Jan. 1988.

[5] L. Bosack and C. Hedrick. Problems in Large LANs. *IEEE Network Magazine*, 2(1), Jan. 1988.

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

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

[8] J. Hart. Extending the IEEE 802.1 MAC Bridge Standard to Remote Bridges. *IEEE Network Magazine*, 2(1), Jan. 1988.

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

[10] T. Jeffree. Unifying Class of Service Provision in LANs - The Role of MAC Bridges. IEE Colloquium, 1999.

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

[12] K. Lui and W. Lee. Spanning tree alternate routing bridge protocol. Technical report, Department of Computer Science, University of Illinois at Urbana-Champaign, Apr. 2001.

[13] R. Perlman. *Interconnections - Bridges, Routers, Switches, and Internetworking Protocols*. Addison-Wesley, second edition, 1999.

[14] R. Perlman, A. Harvey, and G. Varghese. Choosing the Appropriate ISO Layer for LAN Interconnection. *IEEE Network Magazine*, 2(1), Jan. 1988.

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

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

[17] B. Rose. Home networks: a standards perspective. *IEEE Communications Magazine*, 39, Dec. 2001.

[18] W. Seifert. Bridges and Routers. *IEEE Network Magazine*, 2(1), Jan. 1988.

[19] M. Steenstrup. *Routing in Communications Networks*. Prentice-Hall, 1995.

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

[21] G. Varghese and R. Perlman. Transparent Interconnection of Incompatible Local Area Networks Using Bridges. *IEEE Journal on Selected Areas in Communications*, 8(1), Jan. 1990.

[22] XILINK. *FPGA Enabled Home Networking Technology Bridges - Connecting Disparate Technologies*, Mar. 2001.