

A New Approach to Developing High-Availability Server

James T. Yu, Ph.D.

School of Computer Science, Telecommunications, and Information Systems

DePaul University

jyu@cs.depaul.edu

ABSTRACT

This paper presents a new approach to developing high availability (HA) server using Rapid Spanning Tree Algorithm and Protocol (RSTP) as specified in the IEEE 802.1w standard. The traditional approach requires the use of multiple IP addresses and complex management software to monitor multiple links. If there is a link failure, the management software requires to perform complex fail-over functions, including IP address binding, maintenance of stability and integrity of active applications, and synchronization of ARP tables on all clients. This new approach, built on RSTP, is 100% transparent to applications and clients, and does not require any management software. Most importantly, the fail-over time and fall-back time is less than one second. We conducted extensive STP and RSTP experiments in the lab to prove the concept and demonstrate the functionality, and ran an RSTP simulation to demonstrate its performance (measured in fail-over time and fall-back time) for several failure scenarios.

Keywords: High Availability, Spanning Tree Algorithm and Protocol, STP, RSTP, Linux

1. INTRODUCTION

As more and more mission critical applications are deployed over the network, there is a growing demand for high availability (HA) servers. The characteristics of HA servers are no single point of failures and extremely fast fail-over time. The common notion of 5-9's (99.999%) availability translates to less than 5 minutes of downtime per year. In general, an HA server platform has two components, *application* modules and *network* modules. The approach to the application module is to implement redundant processors and data stores; when the primary one fails, the back-up takes over immediately without loss of data and services [1][2][3]. The approach to the network module is to implement redundant network interface cards (NIC) on each server. When the primary interface fails, the back-up interface takes over immediately, without loss of network connectivity. The focus of this study is on the network module.

The current approach to redundant NICs on HA servers is illustrated in Figure 1 where a server is connected to two separated Ethernet switches to avoid any single point of network failures. The dual links between the Ethernet switches run IEEE 802.3ad, also known as Link Aggregation, to prevent link failure between the switches. Without this protection, a link failure would bring half of the clients out of service

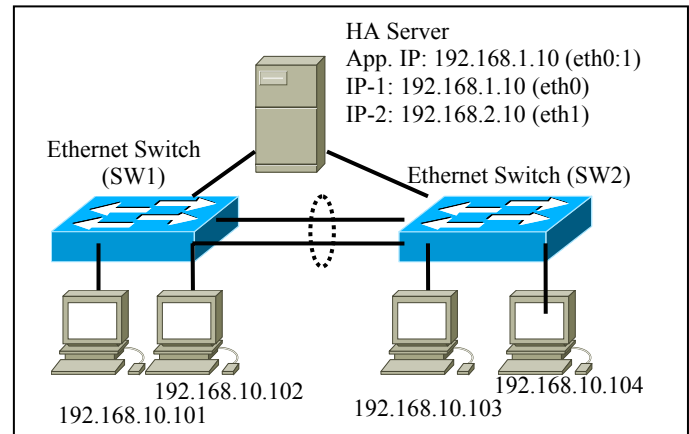


Figure 1. Current HA Server Network Architecture

This solution of Figure 1 requires a complex IP address scheme where each interface is given a *management* IP address (IP-1 and IP-2), and the *application* IP address is bound to one of the interfaces. When the primary interface fails (the interface card, the port, the link, or the switch), the backup interface takes over and the application IP address is switched to the backup interface. An example of the Linux script to perform the fail-over task is given as follows:

```
ifconfig eth0:1 down      #eth0 is primary and eth1 is backup  
ifconfig eth1:1 inet 192.168.10.10 netmask 255.255.255.0 up
```

In order to detect a network failure, the server needs to run a *heartbeat* program to constantly monitor the link integrity to each Ethernet switch. Although this solution offers a high availability platform, it has several major issues:

- The heartbeat program is usually based on the ICMP message (also known as ping), and the time interval is usually set in *seconds*. As a result, this scheme takes more than 10 seconds detecting and recovering from a network link failure[4]. This fail-over time is acceptable to most data applications, but not acceptable to real-time applications, such as Voice over IP (VoIP) where fail-over time over 3 seconds would kill an active voice session.
- The new interface (eth1:1) requires new ARP (Address Resolution Protocol) which is a mapping between the IP address and the MAC address. Unfortunately, the ARP information is on individual client workstations. As a result, when the fail-over occurs, the server needs to inform all clients to *flush* its ARP table and learn the new MAC

address. If this procedure is not performed properly, it could take a few minutes for a client to update its ARP table.

- When the server switches the interface (from eth0:1 to eth1:1), it also needs to *bind* the application IP address to the new interface. This process is transparent to the user workstation as the application IP address stays the same. However, the binding process will kill all existing *sockets*, which kills all active applications. Some applications, such as HTTP and TELNET, can reestablish a new connection (new socket) automatically, but some other applications, such as FTP, may require users to restart it manually.

Equipment vendors are aware of these problems and many have proprietary solutions to address these issues [4]. However, these proprietary solutions are complex and expensive. In addition, one vendor's solution cannot work with another vendor's device (an interoperability issue). This paper presents a simple, but elegant, solution to address and recover from any network failures. Our solution is based on the new IEEE Rapid Spanning Tree Algorithm and Protocol (RSTP) specified in 802.1w [5]. Currently, both Linux and Windows operating systems support STP (802.1D), and they can be enhanced to support the new protocol. Because our solution is based on a standard protocol, there are no issues of interoperability between different vendors' products. The only requirement is the support of the standard protocol, RSTP or STP.

This paper first gives a brief introduction of the STP and RSTP protocols, and then describes how to design an HA server platform using RSTP. We conducted lab experiments using STP (on server) and RSTP (on the switches) to prove the concept from a functional perspective and applied a simulation tool to demonstrate the performance (measured in fail-over time and fall-back time) in sub-seconds. Lastly, we summarize our findings and provide a few directions for future work in this area.

2. ETHERNET AND SPANNING TREE ALGORITHM AND APROTOCOL (STP)

2.1 IEEE 802.1D (STP)

Ethernet, as specified in IEEE 802.3, is the most popular network technology for Local Area Networks (LAN). The original standard is based on Carrier Sense Multiple Access with Collision Detection (CSMA/CD). A major issue of CSMA/CD is with the collision domain where only one station on the domain could send data to the network. A major improvement to Ethernet is the introduction of *switched*^a Ethernet as specified in IEEE 802.1D [7]. In a switched network, there are no collisions and all ports of a network segment can send and receive data at the same

^a The standard terminology is *bridged* network, but it is commonly referenced as *switched* network in the industry. Traditionally, a switch is a multi-port bridge and a bridge is a two-port switch. In the case of HA configuration, a server has two *bridged* ports.

time. One issue with switched Ethernet is the *loop* topology as illustrated in **Figure 2**.

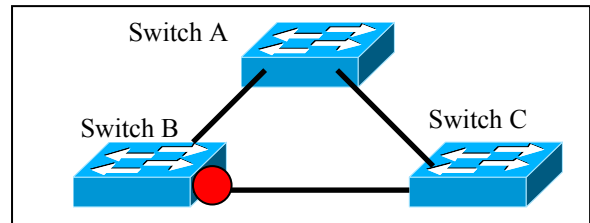


Figure 2. Loop Topology of Ethernet

This loop causes a problem in the broadcast domain. When Switch-A gets a broadcast message, it forwards the message to both Switch-B and Switch-C. Switch-B then forwards the message to Switch-C which also forwards the message back to Switch-A. In the same way, Switch-C gets the message, forward to Switch-B which then forwards back to Switch-A. As a result, the broadcast messages loop around the three Ethernet switches and flood the network with duplicate frames very quickly.

The early approach to this problem is known as Reliable Broadcast Protocol (RBP) [6], and the standard solution to this loop problem is the Spanning Tree Algorithm and Protocol (STP) as specified in IEEE 802.1D[7]. STP converts the physical loop topology into a *logical* tree topology by blocking certain ports on the Ethernet switches. For example, if we block the port on Switch-B going to Switch-C (the port with a red dot on Figure 2), there will be no loop in this network. Each Ethernet port on the switch is either in the **enabled** or **disabled** state. If a port is in the enabled state and STP is turned on, the port is in one of the STP states illustrated in Figure 3.

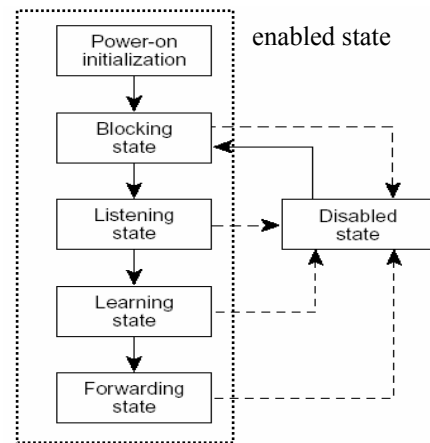


Figure 3. State Transition of STP

The STP protocol specifies three timers to facilitate the state transitions, and these timers are important to measure the fail-over time.

- **Forward delay timer** - the time spent in each of the Listening and Learning states before the Forwarding

state is entered. The min, max, and defaults values are 4, 30, and 15 seconds.

- **Hello timer** – The time interval of a hello packet sent out by the switch. Hello packets are used to communicate information about the topology of the entire bridged LAN. The min, max, and default values are 1, 10, 2 seconds.
- **Maximum [message] age timer** - If the last seen (received) hello packet is older than this timer, it is considered a topology change (link failure). STP shall be recalculated again. The min, max, and default values are 6, 40, and 20 seconds.

STP provides a fault tolerant scheme for the switched Ethernet network as it offers redundant communication paths between any pair of end-points. In the example of **Figure 2**, if the link between Switch-A and Switch-B goes down, Switch-B will go through the STP recalculation and its blocked port will then be in the forwarding state. As a result, the traffic will be resumed after the completion of STP recalculation. Based on the STP state transition and timers, the estimated fail-over time for a simple network like **Figure 2** would be

$$\text{Max Age Timer} + 2 \times \text{Forward Delay Timer}$$

If we use the default values for all timers, the fail-over time would be $20 + 15 \times 2 = 50$ seconds. If we set all timers to its min values, the fail-over time would be $6 + 4 \times 2 = 14$ seconds. If there are multiple switches between the failed link and the root switch, the fail-over time would be much longer. When the failed link is reconnected, STP is recalculated again. During the recalculation, the network is out of service and no user traffic is allowed. This is known as the *fall-back* time. In most cases, fall-back time is about the same as fail-over time. In addition to the long fail-over time (and fall-back time), there is another major issue with STP. During the recalculation of STP, all ports are put into the **blocking** state which is a total network outage. For example, testing, maintenance, or new installation of switches in a wiring closet could bring down the whole campus network for a few minutes. This effect is not acceptable to network administration and management.

2.2. Rapid STP (IEEE 802.1w)

The problem with STP is well known, and we have seen proprietary solutions to this problem [8][9]. An issue with proprietary solutions is that they do not interwork between network devices from different vendors. To address this issue, the IEEE standard committee introduced a new standard, Rapid Spanning Tree Algorithm and Protocol (RSTP), and it is referenced as IEEE 802.1w. There are several improvements of RSTP over STP:

- RSTP maintains the concept of the blocking and forwarding states, and keeps the learning state. It does not have the listening state. As a result, RSTP could save at

least one cycle of Forward Delay Time during the fail-over process.

- A new feature in RSTP is to assign each port a *role*: including root port, designated port, alternate port, and backup port. At the steady state, the root port and designated port are in the **forwarding** state, and the alternate port and the backup port are in the **blocking** state. When a link failure is detected, an alternate port is moved to the forwarding state *immediately* so the traffic is also resumed immediately. This feature significantly reduces the fail-over time to sub-seconds in many failure scenarios.
- When a link failure is detected, the switch being affected assumes the role of the root switch and continues forwarding frames based on this new role. During the STP recalculation, user traffic is still carried on the network. This feature addresses the issue of global effect of STP recalculation and avoids the total network outage.

Most vendors claim that their RSTP-enabled switch products can achieve fail-over time in less than one second, and our lab experiments confirm it. (See Section 4.)

3. NEW APPROACH TO HIGH AVAILABILITY LINUX SERVER

As discussed earlier, STP provides a fail-over scheme at the Data Link layer, and our approach is to apply this protocol to develop high availability server. Both Linux and Windows operating systems have built-in Ethernet bridge capability [10][11], and both support Spanning Tree Algorithm and Protocol (STP) as specified in IEEE 802.1D.^b If we configure a Linux or Windows server as an Ethernet bridge, it will be able to detect network and link failures automatically using STP. There is no need for additional management software on the server. The physical architecture is the same, but the IP address scheme is much simpler as illustrated in Figure 4.

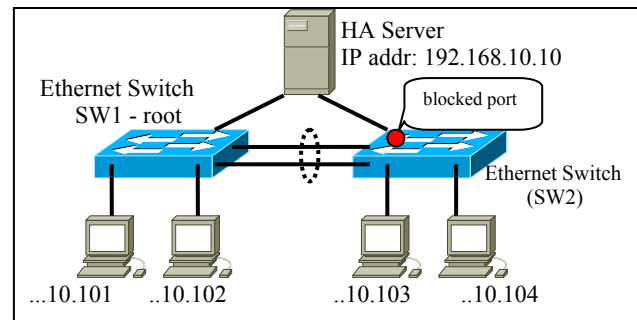


Figure 4. New HA Architecture using STP

The configuration procedure for Linux bridge is given in the following script:

^b Linux provides a utility (**brctl**) to configure STP parameters. Windows does not have such a utility for STP configuration. As a result, our experiments in this study are conducted on Linux.

```

brctl addbr br1          # Create a bridge interface
brctl addif br1 eth0     # Add interface to the bridge
brctl addif br1 eth1
ifconfig eth0 0.0.0.0    # Reset IP address
ifconfig eth1 0.0.0.0
ifconfig br1 up         # Bring up the bridge
ifconfig br1 192.168.10.10 netmask 255.255.255.0 up # set bridge IP
route add default gw 192.168.10.1 # set a default gateway

```

In this new architecture, the Linux server is configured as an Ethernet bridge, and it runs STP with the other two Ethernet switches. STP transforms a loop topology to a loop-free topology (i.e., a tree topology) by blocking certain ports so that broadcast messages would not loop around the network. This technique provides a fault tolerant scheme for Ethernet networks. STP has two *steady* states for each port: **forwarding** and **blocking**. One port on the Ethernet switch shall be in the **blocking** state to prevent loops of the network. When there is a link failure, STP will detect it and change the port state from **blocking** to **forwarding**. With STP configured on the Linux server, there is no need for additional HA software. In addition, it uses a single IP address and does not need to bind (or rebind) during the fail-over process. The solution is *theoretically* sound but has two *practical* issues: long fail-over time and global effect of STP recalculation as presented in Section 3.1.

To address the issues of STP, we propose to use Rapid Spanning Tree Algorithm and Protocol (RSTP) as specified in IEEE 802.1w. RSTP has local *intelligence* so that it knows how to move a port from the **blocking** state to the **forwarding** state *immediately* while maintaining a tree topology. The network architecture and IP address assignment are the same for STP and RSTP configurations.

4. EXPERIMENTS

IEEE 802.1w (RSTP) is a relatively new standard. Most vendors do not offer this feature in their Ethernet products until early 2003. Our experiment uses two Cisco Catalyst 2950 switches with enhanced image (EI) which supports RSTP. As of the writing of this paper, we find only STP on Linux where RSTP is not available yet. As a result, our experiment is based on the combination of STP (on Linux) and RSTP (on Catalyst 2950). The network configuration is illustrated in Figure 5. The label on each link represents a failure scenario where we unplug the labeled cable.

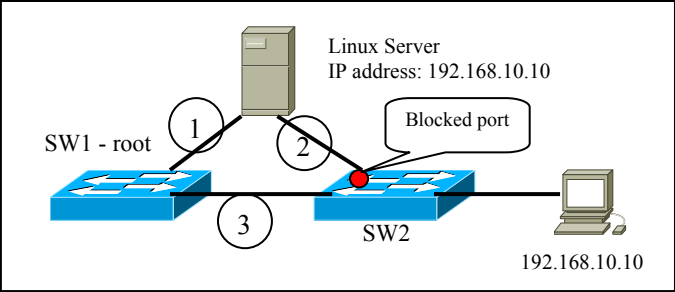


Figure 5. STP and RSTP Lab Experiment

The user workstation continues **pinging** the Linux server every second. In a failure scenario, the number of ICMP packets lost is the fail-over time in seconds. We studied two configurations on Linux, one with STP on and the other with STP off. When STP is on, we configure the switch priority and path cost so that the **blocked** port is on SW2 as shown in Figure 5. When STP is off on Linux, both Linux ports are in the **forwarding** state. We collected fail-over and fall-back data of different STP timers. The results of minimal STP timers are summarized in Table 1.

Table 1. RSTP Experiment Results of HA Linux Server (Hello Time: 1 sec. Forward Delay: 4 sec. Max Age: 6 sec)

Case ID	Configuration	Failure Scenario	Fail-over	Fall-back
1	STP up on Linux	Link failure on active port (1)	14	6
2		Link failure on blocked port (2)	0	0
3		Link failure b/w switches (3)	<1	<1
4	STP off on Linux	Link failure on active port (1)	16	16
5		Link failure on blocked ports (2)	0	0
6		Link failure b/w switches (3)	<1	<1

The experiment results show that RSTP could yield satisfactory results among switches supporting RSTP (cases 3&6). However, RSTP does not work well with STP where the results are determined by STP, and not by RSTP. The experiment, however, does prove the concept and functionality of the proposed architecture, and it shows a simple and cost-effective mechanism for protecting network failures. We conducted another experiment using an Ethernet *hub* to connect both Catalyst switches and the Linux server [12]. The results are similar to the case of disabling STP on Linux. The reason is that an RSTP port has to work with another RSTP port; otherwise, RSTP resorts back to STP.

5. SIMULATION

Although we do not find RSTP software on Linux, we find a software product that simulates RSTP on Linux [13]. This software has two modules: *manager* and *bridge* where the bridge simulates a 4-port Ethernet bridge. We modified the source code of the bridge module and created two new modules, *server* and *client*. The server module has two ports to simulate an RSTP-enabled bridge. The client module has only one port and this port should always be in the **forwarding** state even during the phase of STP recalculation. We use this simulation software to create a network similar to Figure 5 and it is illustrated in Figure 6. The next step is to use the *manager* module to create different failure scenarios. In all the failures cases, we were able to observe fail-over time in less than one second which is demonstrated by the transition time from the **blocking** state to the **forwarding** state.

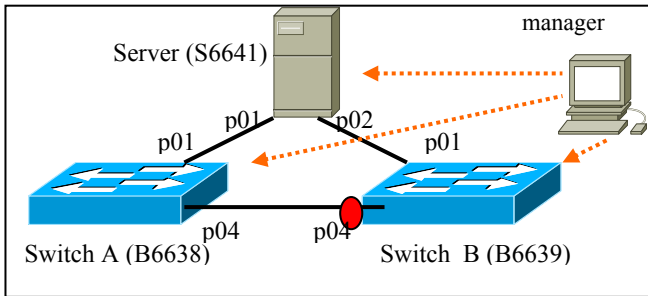


Figure 6. Simulation of RSTP-based HA Server Platform

The initial configuration of the server and two switches (bridges) are given in Table 2.

Table 2. Initial RSTP Configurations on the Switches and Server

	Port	State	Role
Server (ID: S6641)	p01	FWD	Root
	p02	FWD	Desig
Switch (ID: B6638)	p01	FWD	Desig
	p04	FWD	Desig
Switch (ID: B6639)	p01	FWD	Root
	p04	BLK	ALT

We then used the manager station to *disconnect* the link between the server and Switch. As shown in Table 2, the blocked port (p04 on Switch B) had the **alternate** role and it should be transitioned to the forwarding state immediately. We collected the time log of RSTP state transition on Switch-B (B6639) and the results confirm our understanding of RSTP operation. (The first part of the RSTP log is the time stamp.) Note that the transition from the blocked state to the forwarding state still goes through the learning state, but this can be done in less than one second. We should note that the port-p01 on Switch B also goes through the complete cycle of state transition where the cycle is completed in less than one second.

Time Log of Station Transition of Switch B (ID=B6639)

```
14:49:15: (p01): FORWARDING=>DISCARDING*
14:49:15: (p04): DISCARDING=>LEARNING
14:49:15: (p04): LEARNING=>FORWARDING
14:49:15: (p01): DISCARDING=>LEARNING
14:49:15: (p01): LEARNING=>FORWARDING
```

*The discarding state (of this simulation tool) is the same as the disabled state in Figure 3.

After the recalculation of RSTP, we observed the following configurations.

Table 3. RSTP Configuration after the Fail-Over Process

	Port	State	Role
Server (ID: S6641)	p01	Discard	
	p02	FWD	Desig
Switch (ID: B6638)	p01	Discard	
	p04	FWD	Desig
Switch (ID: B6639)	p01	FWD	Root
	p04	FWD	Desig

When we connected the link back (the fall-back test), we observe the following state transitions on the switches and the server.

Server (S6641)	
15:19:59: (p02):	FORWARD => DISCARD
15:19:59: (p01):	DISCARD=>LEARNING
15:19:59: (p01):	LEARNING=>FORWARD
15:19:59: (p02):	DISCARD=>LEARNING
15:19:59: (-p02):	LEARNING=>FORWARD
Switch A (B6638)	
15:19:59: (p01):	DISCARD=>LEARNING
15:19:59: (p01):	LEARNING=>FORWARD
Switch B (6639)	
15:19:59: (p04):	FORWARD=>DISCARDING
15:19:59: (p04):	DISCARD=>BLOCK

This simulation results confirm our understanding of RSTP and its application to design a high availability server. In all the failure scenarios, we are able to observe the fail-over time and fall-back time within one second which is a significant improvement to most of the existing approach to high availability server platform.

6. CONCLUSION AND FUTURE WORK

This paper presents a new and cost-effective solution to develop a high-availability server platform. Our experiment results prove that the concept is theoretically sound and functionally working, and our simulation results show that the fail-over time and fall-back time are satisfactory. Although Linux and Windows do not support RSTP at the present time, we expect this issue to be resolved quickly. The advantages of this RSTP-based HA server platform are summarized as follows:

- This fail-over due to network failures is transparent to the server. If there is a network failure (card, link, port, or switch), the traffic is routed to the backup interface. Server does not need to do anything, so there is no need for costly management software to monitor the network.
- The binding of IP address and MAC address stays the same. This may be the most significant benefit of this approach, and it implies the fail-over procedure is transparent to client workstations.
- This is a layer-2 solution, which is transparent to upper layer functions. For example, the IP layer maintains the same *socket* during failover, so there is no interruption to any services.
- This is a cost-effective solution to address network failures because the software is in the Linux kernel. There is no additional software needed for this implementation.
- Because of the fast failover time, RSTP could work seamlessly with upper layer protocols to provide a complete fault tolerant solution. For example, we can run RSTP at layer 2 and VRRP (Virtual Redundant Routing Protocol) at layer 3 on the same server [14]. It takes 3-5 seconds to recover from a typical VRRP failure, and RSTP takes less than one second. The faster failover scheme at the lower layer prevents any cyclic effect.

In addition to high availability, we are also interested in *load balancing* to increase the overall network throughput by 100%. We are studying a scheme similar to Link Aggregation as specified in IEEE 802.3ad which requires aggregated links to be terminated on the same switch. The basic approach is to creating independent user sessions so that traffic can be routed to both ports on the Linux server. We understand that there are many complex issues, such as MAC address forwarding on the Ethernet switch and ARP on the client stations. We hope our research could stimulate more interests in addressing these challenging issues.

REFERENCES

- [1] High Availability Linux Project Web Site, <http://www.linux-ha.org/>
- [2] Alan Wood, "Predicting Client/Server Availability," *IEEE Computer*, April 1985, pp. 41-48.
- [3] A. K. Somani and N. H. Vaidya, "Understanding Fault Tolerance and Reliability," *IEEE Computer*, April 1997, pp. 45-50.
- [4] A. Bhutani and Z. Mahmood, "Using NIC Teaming to Achieve High Availability on Linux Platforms," *Dells Magazine*, February 2003. <http://ftp.us.dell.com/app/1q03-Bhu.pdf>
- [5] MAC Bridges – Rapid Reconfiguration, IEEE 802.1w, 2001
- [6] D. Venkatesuli and T. A. Gonsalves, "Efficient Fault-Tolerant Reliable Broadcast in an Extended :LAN," Proceedings of the 22nd IEEE Conference on Local Computer Networks 1997.
- [7] Medium Access Control (MAC) Bridges, IEEE 802.1D, 1998.
- [8] Cisco Uplink Fast Feature - <http://www.cisco.com/warp/customers/473/51.html>
- [9] Hirschmann Electronics, Inc. Distributed Communication Architecture: Industrial Networking Solutions with a Future. A white paper at COMDEX 2000
- [10] James T. Yu, "Performance Evaluation on Linux Bridge," Telecommunications System Management Conference 2004, Louisville, Kentucky (April 2004)
- [11] Chibiao Liu and James T. Yu, "Applications and Performance Analysis of Bridging with L3 Forwarding on Wireless LANs," CTI Research Symposium, DePaul University, (November 2004)
- [12] James T. Yu, "Applying IEEE 802.1w (RSTP) to Improve Service Availability," *IEEE International Conference on Dependable Systems and Networks*, (June 2003), pp. B10-B11.
- [13] RSTP 802.1w Library & Simulator. <http://rstplib.sourceforge.net/>
- [14] IETF RFC 2338, Virtual Router Redundancy Protocol, April 1998.