Enterprise Network Design Patterns: High Availability

Deepak Kakadia, Sun Microsystems, Inc.

Sam Halabi and Bill Cormier, Extreme Networks, Inc.

Sun BluePrints™ OnLine—September 2002

http://www.sun.com/blueprints

Sun Microsystems, Inc.
4150 Network Circle
Santa Clara, CA 95045 U.S.A.
650 960-1300

Part No. 816-7883-10
Revision A, 09/09/02
Edition: September 2002

Please
Recycle

Adobe PostScript

Enterprise Network Design Patterns: High Availability

Availability has always been an important design goal for network architectures. As enterprise customers increasingly deploy mission-critical web-based services, they require a deeper understanding of designing optimal network availability solutions. There are several approaches to implementing high-availability network solutions. This article provides an overview of the various approaches and describes where it makes sense to apply that solution.

FIGURE 1 provides a high-level overview of a typical corporate customer's network. This integrated network can be divided into the following sectors to create a logical partitioning, which can be helpful in understanding the motivation of the protocols that provide resiliency.

- Access network—This sector connects the enterprise's private network to the service provider. This network is generally controlled by a network service provider, which is usually called an Internet service provider (ISP) because that provider provides connectivity to the Internet. The term *access network* is used by carriers because this is the point where end users and enterprises access the carrier networks. Depending on the configuration, there may be a static route from the enterprise to the ISP, or there may be an exterior routing protocol such as Border Gateway Protocol4 (BGP4). BGP4 is more resilient, if a particular route is down, an alternate route may be available.
- Enterprise network—This network is the enterprise's internal network, which is always partitioned and segregated from the external network primarily for security reasons. This network is the focus of our paper. Several methods provide network resiliency which we investigate further in this article.
- Corporate WAN—These networks provide the connectivity over long distances to the remote enterprise sites. There are varying degrees of connectivity, which include campus networks, that interconnect enterprise buildings within a certain distance: metropolitan area networks (MANs) that interconnect enterprise offices located within one local providers MAN network, and wide area networks (WANs) that connect enterprise branch offices that may span thousands of miles.

WAN connectivity generally requires the services of a Tier 1 service provider. Modern WAN providers may provide an IP tunnel for that enterprise to connect remote offices over a shared network.

In this paper, we briefly discuss how MPLS can be used for resiliency. MPLS has gained wide industry acceptance in the core networks.

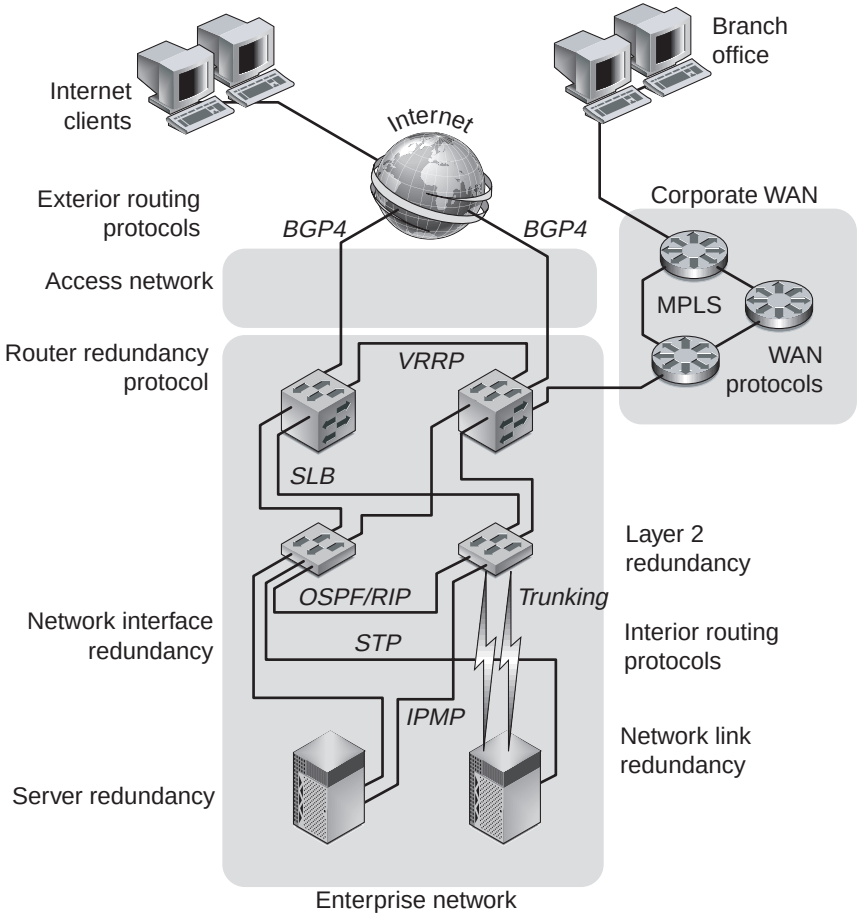

FIGURE 1 Networking Features to Increase Availability

The scope of this article is limited to interior routing protocols and enterprise network technologies for availability purposes.

Physical Network Topology and Availability

One of the first items to consider for network availability is the physical topology from an implementation perspective. In general, the topology will have a direct impact on the mean time between failure (MTBF) calculation. Serial components reduce availability and parallel components increase availability.

There are three topology aspects impacting network availability:

- Component failure—This aspect is the probability of the device failing. It is measured using statistics averaging the amount of time the device works divided by the average time the device works plus the failed time. This value is called the MTBF. In calculating the MTBF, components that are connected serially drastically reduce the MTBF, while components that are in parallel, increase the MTBF.

  FIGURE 2 shows two network designs. In both designs, Layer 2 switches simply provide physical connectivity for one virtual local area network (VLAN) domain. Layer 2-7 switches are multilayer devices providing routing, load balancing, and other IP services, in addition to physical connectivity.

  Design A shows a flat architecture, often seen with multi-layer chassis based switches using Extreme Networks Black Diamond®, Foundry Networks BigIron®, or Cisco® switches. The switch can be partitioned into VLANs, isolating traffic from one segment to another, yet providing a much better solution overall. In this approach, the availability will be relatively high, because there are two parallel paths from the ingress to each server and only two serial components that a packet must traverse in order to reach the target server.

  In Design B, the architecture provides the same functionality, but across many small switches. From an availability perspective, this solution will have a relatively lower MTBF because of the fact there are more serial components that a packet must traverse in order to reach a target server. Other disadvantages of this approach include manageability, scalability, and performance. However, one can argue that there may be increased security using this approach, which in some customer requirements, outweighs all other factors. In Design B, multiple switches need to be hacked to control the network; whereas in Design A, only one switch needs to be hacked to bring down the entire network.

- System failure—This aspect captures failures that are caused by external factors, such as a technician accidentally pulling out a cable. The more components that are potential candidates for failure are directly proportional to the complexity, and thus, result in a higher system failure probability. So Design B, in FIGURE 2, has more components that can go wrong, which contributes to the increased probability of failure.

- Single points of failure—This aspect captures the number of devices that can fail and still have the system functioning. Both approaches have no single points of failure, and are equal in this regard. However, Design B is somewhat more resilient because if a network interface card (NIC) fails, that failure is isolated by the Layer 2 switch, and does not impact the rest of the architecture. This issue is a trade-off to consider, where availability is sacrificed for increased resiliency and isolation of failures.

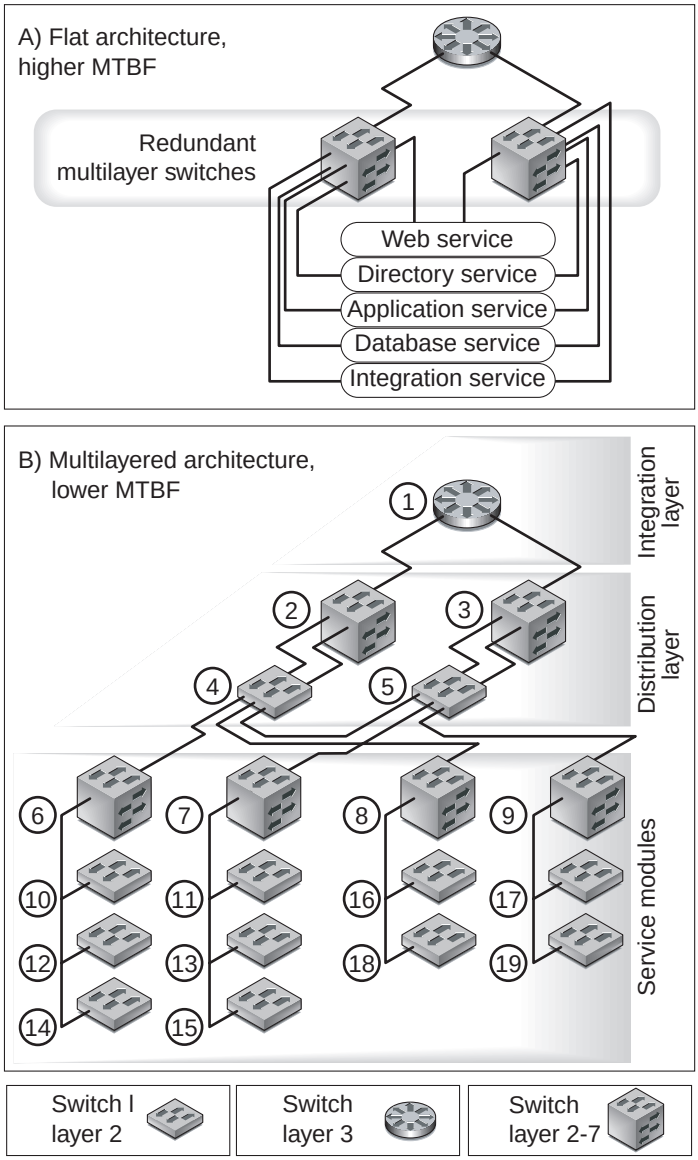

FIGURE 2 Network Topologies and Impact on Availability

Layer 2 Availability: Trunking —802.3ad—Link Aggregation

Link aggregation or trunking increases availability by distributing network traffic over multiple physical links. If one link breaks, the load on the broken link is transferred to the remaining links.

IEEE 802.3ad is an industry standard created to allow the trunking solutions of various vendors to interoperate. Like most standards, there are many ways to implement the specifications. Link aggregation can be thought of as a layer of indirection between the MAC and PHY layer. Instead of having one fixed MAC address that is bound to a physical port, a logical MAC address is exposed to the IP layer and implements the Data Link Provider Interface (DLPI). This logical MAC address can be bound to many physical ports. The remote side must have the same capabilities and algorithm for distributing packets among the physical ports. FIGURE 3 shows a breakdown of the sub-layers.

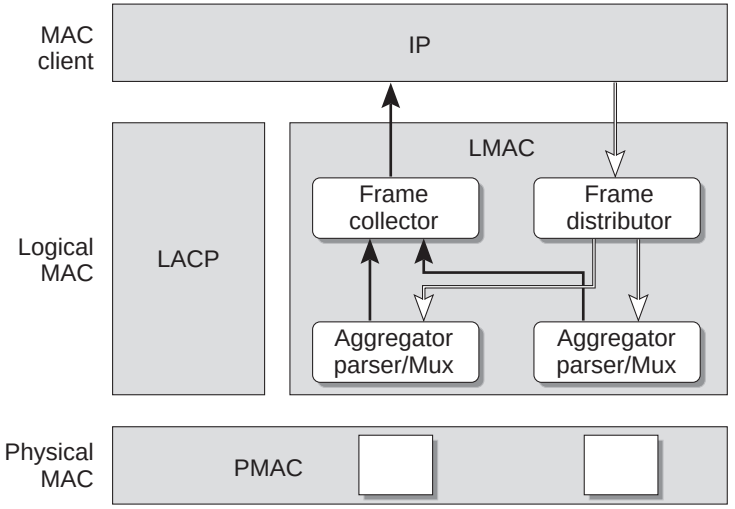

FIGURE 3 Trunking Software Architecture

Theory of Operation

The Link Aggregation Control Protocol (LACP), allows both ends of the trunk to communicate trunking or link aggregation information. The first command that is sent is the `Query` command, where each link partner discovers the link aggregation capabilities of each other. If both partners are willing and capable, a `Start Group` command is sent, which indicates that a link aggregation group is to be created; followed by adding segments to this group, which includes link identifiers tied to the ports participating in the aggregation.

The LACP can also delete a link, which may be due to the detection of a failed link. Instead of balancing the load across the remaining ports, the algorithm simply places the failed links traffic onto one of the remaining links. The collector reassembles traffic coming from the different links. The distributor, takes an input stream and spreads out the traffic across the ports belonging to a trunk group or link aggregation group.

Availability Issues

To understand suitability for network availability, Sun Trunking™ 1.2 software was installed on several quad fast Ethernet cards. The client has four trunks connected to the switch. The server also has four links connected to the switch. This setup allows the load to be distributed across the four links, as shown in FIGURE 4.

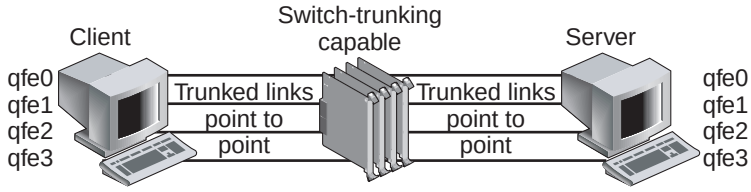

FIGURE 4 Trunking Failover Test Setup

The highlighted (in *bold italic*) line in the following output shows the traffic from the client `qfe0` moved to the server `qfe1` under load balancing.

```
Jan 10 14:22:05 2002

Name   Ipkts        Ierrs  Opkts        Oerrs  Collis  Crc     %Ipkts  %Opkts

qfe0   210          0      130          0      0       0       100.00   25.00
qfe1   0            0      130          0      0       0         0.00   25.00
qfe2   0            0      130          0      0       0         0.00   25.00
```

```
Jan 10 14:22:05 2002
qfe3   0            0      130          0      0       0         0.00   25.00

(Aggregate Throughput(Mb/sec):     5.73(New Peak)    31.51(Past Peak)
18.18%(New/Past))


Jan 10 14:22:06 2002

Name   Ipkts        Ierrs  Opkts        Oerrs  Collis  Crc     %Ipkts  %Opkts

qfe0   0            0      0            0      0       0         0.00    0.00
qfe1   0            0      0            0      0       0         0.00    0.00
qfe2   0            0      0            0      0       0         0.00    0.00
qfe3   0            0      0            0      0       0         0.00    0.00

(Aggregate Throughput(Mb/sec):     0.00(New Peak)    31.51(Past Peak)
0.00%(New/Past))


Jan 10 14:22:07 2002

Name   Ipkts        Ierrs  Opkts        Oerrs  Collis  Crc     %Ipkts  %Opkts

qfe0   0            0      0            0      0       0         0.00    0.00
qfe1   0            0      0            0      0       0         0.00    0.00
qfe2   0            0      0            0      0       0         0.00    0.00
qfe3   0            0      0            0      0       0         0.00    0.00

(Aggregate Throughput(Mb/sec):     0.00(New Peak)    31.51(Past Peak)
0.00%(New/Past))


Jan 10 14:22:08 2002

Name   Ipkts        Ierrs  Opkts        Oerrs  Collis  Crc     %Ipkts  %Opkts

qfe0   0            0      0            0      0       0         0.00    0.00
qfe1   1028         0      1105         0      0       0       100.00   51.52
qfe2   0            0      520          0      0       0         0.00   24.24
qfe3   0            0      520          0      0       0         0.00   24.24

(Aggregate Throughput(Mb/sec):    23.70(New Peak)    31.51(Past Peak)
75.21%(New/Past))
```

Several test transmission control protocol (TTCP) streams were pumped from one host to the other. When all links were up, the load was balanced evenly and each port experienced a 25 percent load. When one link was cut, the traffic of the failed link (qfe0) was transferred onto one of the remaining links (qfe1) which then showed a 51 percent load.

The failover took three seconds. However, if all links were heavily loaded, the algorithm might force one link to be saturated with its original link load in addition to the failed links traffic. For example, if all links were running at 55 percent capacity and one link failed, one link would be saturated at 55 percent + 55 percent = 110 percent traffic. Link aggregation is suitable for point-to-point links for increased availability, where nodes are on the same segment. However, there is a trade-off of port cost on the switch side as well as the host side.

Layer 2 Availability: Spanning Tree Protocol

The spanning tree algorithm was developed by Radia Perlman, currently with Sun Microsystems. The Spanning Tree Protocol (STP) is used on Layer 2 networks to eliminate loops. For added availability, redundant Layer 2 links can be added, however, these redundant links introduce loops, which cause bridges to forward frames indefinitely.

By introducing STP, bridges communicate with each other by sending bridge protocol data units (BPDUs), which contain information that a bridge uses to determine which ports forward traffic and which ports don’t, based on the spanning tree algorithm. A typical BPDU contains information including a unique bridge identifier, the port identifier, and cost to the root bridge, which is the top of the spanning tree.

From these BPDUs, each bridge can compute a spanning tree and decide which ports to direct all forwarding of traffic. If a link fails, this tree is recomputed, and redundant links are activated by turning on certain ports, hence creating increased availability. A network needs to be designed to ensure that every possible link that may fail has some redundant link.

In older networks, bridges are still used. However, with recent advances in network switch technology and smaller Layer 2 networks, bridges are not used as much.

Availability Issues

To better understand failure detection and recovery, a testbed was created, as shown in FIGURE 5.

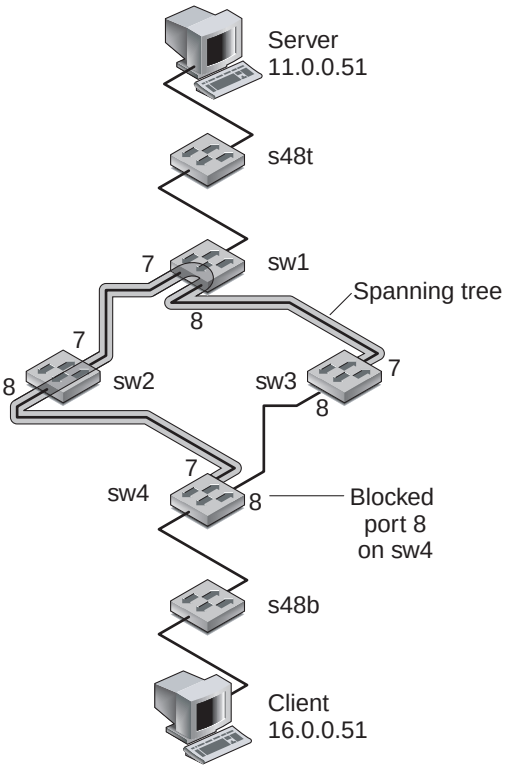

FIGURE 5 Spanning Tree Network Setup

The switches, sw1, sw2, sw3, and sw4, were configured in a Layer 2 network, with an obvious loop, which was controlled by running the STP among these switches. On the client, we ran the `traceroute server` command, resulting in the following output, which shows that the client sees only two Layer 3 networks: the 11.0.0.0 and the 16.0.0.0 network.

```
client># traceroute server
traceroute: Warning: Multiple interfaces found; using 16.0.0.51 @
hme0
traceroute to server (11.0.0.51), 30 hops max, 40 byte packets
 1  16.0.0.1 (16.0.0.1)  1.177 ms  0.524 ms  0.512 ms
 2  16.0.0.1 (16.0.0.1)  0.534 ms !N  0.535 ms !N  0.529 ms !N
```

Similarly, the server sees only two Layer 3 networks, on the server we ran `traceroute client` command and got the following output:

```
server>#  traceroute client
traceroute: Warning: Multiple interfaces found; using 11.0.0.51 @
hme0
traceroute to client (16.0.0.51), 30 hops max, 40 byte packets
 1  11.0.0.1 (11.0.0.1)  0.756 ms  0.527 ms  0.514 ms
 2  11.0.0.1 (11.0.0.1)  0.557 ms !N  0.546 ms !N  0.531 ms !N
```

The following outputs show the STP configuration and port status of the participating switches, showing the root switches port MAC address.

```
* sw1:17 # sh s0 ports     7-8
Stpd: s0       Port: 7 PortId: 4007   Stp: ENABLED   Path Cost: 4
Port State: FORWARDING          Topology Change Ack: FALSE
Port Priority: 16
Designated Root:  80:00:00:01:30:92:3f:00     Designated Cost: 0
Designated Bridge: 80:00:00:01:30:92:3f:00      Designated Port
Id: 4007

Stpd: s0       Port: 8 PortId: 4008   Stp: ENABLED   Path Cost: 4
Port State: FORWARDING          Topology Change Ack: FALSE
Port Priority: 16
Designated Root:  80:00:00:01:30:92:3f:00     Designated Cost: 0
Designated Bridge: 80:00:00:01:30:92:3f:00      Designated Port
Id: 4008
```

```
* sw2:12 # sh s0 ports     7-8
Port Mode   State      Cost   Flags Priority Port ID Designated
Bridge
7    802.1D FORWARDING 4      e-R-- 16       16391
80:00:00:01:30:92:3f:00
8    802.1D FORWARDING 4      e-D-- 16       16392
80:00:00:01:30:92:3f:00

Total Ports: 8

Flags: e=Enable, d=Disable, T=Topology Change Ack
       R=Root Port, D=Designated Port, A=Alternative Port
```

```
* sw3:5 # sh s0 ports     7-8
Stpd: s0       Port: 7 PortId: 4007   Stp: ENABLED   Path Cost: 4
Port State: FORWARDING          Topology Change Ack: FALSE
Port Priority: 16
Designated Root:  80:00:00:01:30:92:3f:00     Designated Cost: 0
Designated Bridge: 80:00:00:01:30:92:3f:00      Designated Port
Id: 4001

Stpd: s0       Port: 8 PortId: 4008   Stp: ENABLED   Path Cost: 4
Port State: FORWARDING          Topology Change Ack: FALSE
Port Priority: 16
Designated Root:  80:00:00:01:30:92:3f:00     Designated Cost: 4
Designated Bridge: 80:00:00:e0:2b:98:96:00      Designated Port
Id: 4008
```

The following output shows that STP has blocked Port 8 on sw4.

```
* sw4:10 # sh s0 ports     7-8
Stpd: s0       Port: 7 PortId: 4007   Stp: ENABLED   Path Cost: 4
Port State: FORWARDING          Topology Change Ack: FALSE
Port Priority: 16
Designated Root:  80:00:00:01:30:92:3f:00     Designated Cost: 4
Designated Bridge: 80:00:00:01:30:f4:16:a0      Designated Port
Id: 4008

Stpd: s0       Port: 8 PortId: 4008   Stp: ENABLED   Path Cost: 4
Port State: BLOCKING            Topology Change Ack: FALSE
Port Priority: 16
Designated Root:  80:00:00:01:30:92:3f:00     Designated Cost: 4
Designated Bridge: 80:00:00:e0:2b:98:96:00      Designated Port
Id: 4008
```

To get a better understanding of failure detection and fault recovery, we conducted a test where the client continually pinged the server, and we pulled a cable on the spanning tree path.

The following output shows that it took approximately 68 seconds for failure detection and recovery, which is not acceptable in most mission-critical environments. (Each ping takes about one second. The following output shows that from `icmp_seq=16` to `icmp_seq=74`, the pings did not succeed.)

```
on client
---------

4 bytes from server (11.0.0.51): icmp_seq=12. time=1. ms
64 bytes from server (11.0.0.51): icmp_seq=13. time=1. ms
64 bytes from server (11.0.0.51): icmp_seq=14. time=1. ms
64 bytes from server (11.0.0.51): icmp_seq=15. time=1. ms
64 bytes from server (11.0.0.51): icmp_seq=16. time=1. ms
ICMP Net Unreachable from gateway 16.0.0.1
 for icmp from client (16.0.0.51) to server (11.0.0.51)
I for icmp from client (16.0.0.51) to server (11.0.0.51)
...
...
ICMP Net Unreachable from gateway 16.0.0.1
 for icmp from client (16.0.0.51) to server (11.0.0.51)
ICMP Net Unreachable from gateway 16.0.0.1
 for icmp from client (16.0.0.51) to server (11.0.0.51)
for icmp from client (16.0.0.51) to server (11.0.0.51)
64 bytes from server (11.0.0.51): icmp_seq=74. time=1. ms
64 bytes from server (11.0.0.51): icmp_seq=75. time=1. ms
64 bytes from server (11.0.0.51): icmp_seq=76.
```

Layer 3—VRRP Router Redundancy

The Virtual Router Redundancy Protocol (VRRP) was designed to remove a single point of failure where hosts connected to the rest of the enterprise network or Internet through one default router. The VRRP is based on an election algorithm, where there are two routers: one master who owns both a MAC and IP address, and a backup, on one LAN or VLAN segment. The hosts all point to one IP address that points to the master router. The master and backup constantly send multicast messages to each other. Depending on the vendor-specific implementation, the backup will assume the master role if the master is no longer functioning or has lowered in priority based on some criteria. The new master also assumes the same MAC address, so that the clients do not need to update their ARP caches.

The VRRP, by itself, has left open many aspects so that switch manufacturers can implement and add features to differentiate themselves. All vendors offer a variety of features that alter the priority, which can be tied to server health checks, number

of active ports, and so on. Whichever router has the highest priority becomes the master. These configurations need to be closely monitored to prevent oscillations. Often, a switch is configured to be too sensitive causing it to constantly change priority, hence, fluctuating from master to backup.

Layer 3—IPMP—Host Network Interface Redundancy

The purpose of the server redundant network interface capability is to increase overall system availability. If one server NIC fails, the backup will take over within two seconds. This is IP Multipathing (IPMP) on the Solaris™ operating environment.

IPMP is a feature bundled with the Solaris operating environment that is crucial in creating highly available network designs. IPMP has a daemon that constantly pings the default router, which is intelligently pulled from the kernel routing tables. If that router is not reachable, another standby interface, in the same IPMP group, then assumes ownership of the floating IP address. The switch then does a re-ARP for the new MAC address and is able to contact the server again.

A typical highly available configuration includes a Sun server that has dual NIC cards, which increases the availability of these components by several orders of magnitude. For example, the GigabitEthernet card, part number 595-5414-01, by itself has an MTBF of 199156 hours, and assuming approximately 2 hours meant time to recovery (MTTR), has an availability of 0.999989958. With two cards, the MTBF becomes 9 9's at .9999999996 availability. This small incremental cost has a big impact on the overall availability computation.

FIGURE 6 shows the Sun server redundant NIC model using IPMP. The server has two NICs, `ge0` and `ge1`, with a fixed IP address of `a.b.c.d` and `e.f.g.h`. The virtual IP address of `w.x.y.z` is the IP address of the service. Client requests use this IP address as the destination. This IP address floats between the two interfaces: `ge0` or `ge1`. Only one interface can be associated with the virtual IP address at any one instant. If the `ge0` interface owns the virtual IP address, then data traffic will follow the P1 path. If the `ge0` interface fails, then the `ge1` interface will takeover and associate the virtual IP address, and then data traffic will follow the P2 path. Failures can be detected within two seconds, depending on the configuration.

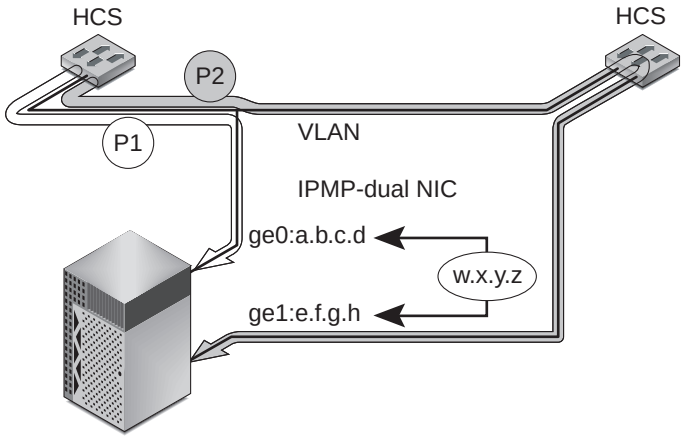

FIGURE 6 High-Availability Network Interface Cards on Sun Servers

Layer 3—Integrated VRRP and IPMP

By combining the availability technologies of routers and server NICs, we can create a reusable cell that can be reused in any deployment where servers are connected to routers. This reusable cell is highly available and scalable. FIGURE 7 shows how this is implemented. Lines 1 and 2 show the VRRP protocol used by the routers to monitor each other. If one router detects that the other has failed, the surviving router assumes the role of master and inherits the IP address and MAC address of the master.

Lines 3 and 5 in FIGURE 7 show how a switch can verify that a particular connection is up and running, which can be por-based, link-based, or Layers 3-, 4-, and 7-based. The router can make synthetic requests to the server and verify that a particular service is up and running. If it detects that the service has failed, then the VRRP can be configured, on some switches, to take this into consideration to impact the election algorithm and tie this failure to the priority of the VRRP router. Simultaneously, the server is also monitoring links. Currently, IPMP consists of a daemon, `in.mpathd`, that constantly pings the default router. As long as the default router can be pinged the master interface (`ge0`) assumes ownership of the IP address. If the `in.mpathd` daemon detects that the default router is not reachable, automatic failover will occur, which brings down the link and floats over the IP address of the server to the surviving interface (`ge1`).

In the lab, we can tune IPMP and Extreme Standby Routing Protocol (ESRP) to achieve failure detection and recovery within one second. The trade-off, is that because the control packets are on the same network as the production network, and

because ESRP is a CPU intensive task, if the switches, networks, or servers become overloaded, false failures are possible because the device can take longer than the strict timeout to respond to the peer's heartbeat.

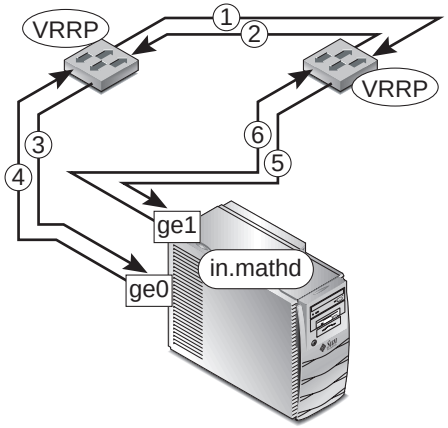

FIGURE 7 Design Pattern—IPMP and VRRP Integrated Availability Solution

Layer 3—OSPF Network Redundancy—Rapid Convergence

Open shortest path first (OSPF) is an intradomain, link-state routing protocol. The main idea of OSPF is that each OSPF router is able to determine the state of the link to all neighbor routers and the associated costs. One property of this routing protocol is that each OSPF router has a view of the entire network, which allows it to find the best path to all participating routers.

All OSPF routers in the domain flood each other with link state packets (LSP), which contain the unique ID of the sending router; a list of directly connected neighbor routers and associated costs; a sequence number and a time to live, authentication, hierarchy, load balancing; and checksum information. From this information, each node can reliably determine if this LSP is the most recent by comparing `seq` number and computing the shortest path to every node, then collecting all LSPs from all nodes and comparing costs, using Dijstras' shortest path algorithm. To prevent continuous flooding, the sender never receives the same LSP packet that he sent out.

To better understand OSPF for suitability from an availability perspective, the following lab network was set up, consisting of Extreme Network switches and Sun servers. FIGURE 8 describes the actual setup used to demonstrate availability characteristics of the interior routing protocol OSPF.

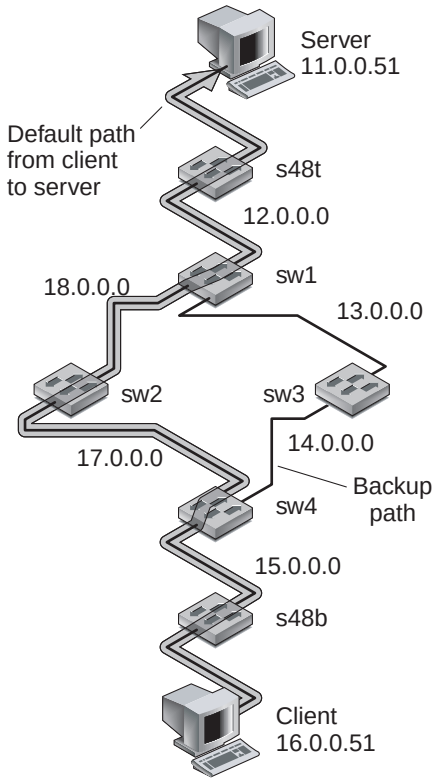

FIGURE 8 Design Pattern—OSPF Network

To confirm correct configuration traceroutes were performed from client to server. In the following output, the highlighted lines show the path through sw2:

```
client># traceroute server
traceroute: Warning: Multiple interfaces found; using 16.0.0.51 @
hme0
traceroute to server (11.0.0.51), 30 hops max, 40 byte packets
 1  16.0.0.1 (16.0.0.1)  1.168 ms  0.661 ms  0.523 ms
 2  15.0.0.1 (15.0.0.1)  1.619 ms  1.104 ms  1.041 ms
 3  17.0.0.1 (17.0.0.1)  1.527 ms  1.197 ms  1.043 ms
 4  18.0.0.1 (18.0.0.1)  1.444 ms  1.208 ms  1.106 ms
 5  12.0.0.1 (12.0.0.1)  1.237 ms  1.274 ms  1.083 ms
 6  server (11.0.0.51)  0.390 ms  0.349 ms  0.340 ms
```

The following tables show the initial routing tables of the core routers. The first two highlighted lines in CODE EXAMPLE 1 show the route to the client through sw2. The second two highlighted lines show the sw2 path.

CODE EXAMPLE 1 Router sw1 Routing Table

```
 OR Destination        Gateway         Mtr   Flags     Use M-Use VLAN     Acct-1
*s  10.100.0.0/24      12.0.0.1          1 UG---S-um    63     0 net12    0
*oa 11.0.0.0/8         12.0.0.1          5 UG-----um    98     0 net12    0
*d  12.0.0.0/8         12.0.0.2          1 U------u-  1057     0 net12    0
*d  13.0.0.0/8         13.0.0.1          1 U------u-    40     0 net13    0
*oa 14.0.0.0/8         13.0.0.2          8 UG-----um     4     0 net13    0
*oa 15.0.0.0/8         18.0.0.2         12 UG-----um     0     0 net18    0
*oa 15.0.0.0/8         13.0.0.2         12 UG-----um     0     0 net13    0
*oa 16.0.0.0/8         18.0.0.2         13 UG-----um     0     0 net18    0
*oa 16.0.0.0/8         13.0.0.2         13 UG-----um     0     0 net13    0
*oa 17.0.0.0/8         18.0.0.2          8 UG-----um     0     0 net18    0
*d  18.0.0.0/8         18.0.0.1          1 U------u-   495     0 net18    0
*d  127.0.0.1/8        127.0.0.1         0 U-H----um     0     0 Default  0

Origin(OR): b - BlackHole, bg - BGP, be - EBGP, bi - IBGP, bo - BOOTP, ct - CBT
            d - Direct, df - DownIF, dv - DVMRP, h - Hardcoded, i - ICMP
           mo - MOSPF, o - OSPF, oa - OSPFIntra, or - OSPFInter, oe - OSPFAsExt
            o1 - OSPFExt1, o2 - OSPFExt2, pd - PIM-DM, ps - PIM-SM, r - RIP
            ra - RtAdvrt, s - Static, sv - SLB_VIP, un - UnKnown.

Flags: U - Up, G - Gateway, H - Host Route, D - Dynamic, R - Modified,
       S - Static, B - BlackHole, u - Unicast, m - Multicast.

Total number of routes = 12.
```

CODE EXAMPLE 1 Router sw1 Routing Table *(Continued)*

```
Mask distribution:
   11 routes at length  8           1 routes at length 24
```

CODE EXAMPLE 2 Router sw2 Routing Table

```
sw2:8 # sh ipr

 OR Destination        Gateway         Mtr   Flags     Use M-Use VLAN     Acct-1
*s  10.100.0.0/24      18.0.0.1          1 UG---S-um    27     0 net18    0
*oa 11.0.0.0/8         18.0.0.1          9 UG-----um    98     0 net18    0
*oa 12.0.0.0/8         18.0.0.1          8 UG-----um     0     0 net18    0
*oa 13.0.0.0/8         18.0.0.1          8 UG-----um     0     0 net18    0
*oa 14.0.0.0/8         17.0.0.2          8 UG-----um     0     0 net17    0
*oa 15.0.0.0/8         17.0.0.2          8 UG-----um     9     0 net17    0
*oa 16.0.0.0/8         17.0.0.2          9 UG-----um     0     0 net17    0
*d  17.0.0.0/8         17.0.0.1          1 U------u-    10     0 net17    0
*d  18.0.0.0/8         18.0.0.2          1 U------u-   403     0 net18    0
*d  127.0.0.1/8        127.0.0.1         0 U-H----um     0     0 Default  0
#
#
```

CODE EXAMPLE 3 Router sw3 Routing Table

```
sw3:5 # sh ipr

 OR Destination        Gateway         Mtr   Flags     Use M-Use VLAN     Acct-1
*s  10.100.0.0/24      13.0.0.1          1 UG---S-um    26     0 net13    0
*oa 11.0.0.0/8         13.0.0.1          9 UG-----um     0     0 net13    0
*oa 12.0.0.0/8         13.0.0.1          8 UG-----um   121     0 net13    0
*d  13.0.0.0/8         13.0.0.2          1 U------u-    28     0 net13    0
*d  14.0.0.0/8         14.0.0.1          1 U------u-    20     0 net14    0
*oa 15.0.0.0/8         14.0.0.2          8 UG-----um     0     0 net14    0
*oa 16.0.0.0/8         14.0.0.2          9 UG-----um     0     0 net14    0
*oa 17.0.0.0/8         14.0.0.2          8 UG-----um     0     0 net14    0
*oa 18.0.0.0/8         13.0.0.1          8 UG-----um     0     0 net13    0
*d  127.0.0.1/8        127.0.0.1         0 U-H----um     0     0 Default  0
```

The first two highlighted lines in CODE EXAMPLE 4 show the route back to the server through sw4, using the first in the table. The second two highlighted lines show the sw2 path.

CODE EXAMPLE 4 Switch sw4 Routing Table

```
sw4:8 # sh ipr

 OR Destination        Gateway         Mtr   Flags     Use M-Use VLAN     Acct-1
*s  10.100.0.0/24      14.0.0.1          1 UG---S-um    29     0 net14    0
*oa 11.0.0.0/8         17.0.0.1         13 UG-----um     0     0 net17    0
*oa 11.0.0.0/8         14.0.0.1         13 UG-----um     0     0 net14    0
*oa 12.0.0.0/8         17.0.0.1         12 UG-----um     0     0 net17    0
*oa 12.0.0.0/8         14.0.0.1         12 UG-----um     0     0 net14    0
*oa 13.0.0.0/8         14.0.0.1          8 UG-----um     0     0 net14    0
*d  14.0.0.0/8         14.0.0.2          1 U------u-    12     0 net14    0
*d  15.0.0.0/8         15.0.0.1          1 U------u-   204     0 net15    0
*oa 16.0.0.0/8         15.0.0.2          5 UG-----um     0     0 net15    0
*d  17.0.0.0/8         17.0.0.2          1 U------u-    11     0 net17    0
*oa 18.0.0.0/8         17.0.0.1          8 UG-----um     0     0 net17    0
*d  127.0.0.1/8        127.0.0.1         0 U-H----um     0     0 Default  0
```

To check failover capabilities on the OSPF, the interface on the switch sw2 was damaged to create a failure and a constant ping was run from the client to the server.

The interface on the switch sw2 was removed, and the measurement of failover was performed as shown in the following output. The first highlighted line shows when the interface sw2 fails. The second highlighted line shows the new switch interface sw3 route is established in two seconds.

```
client reading:

64 bytes from server (11.0.0.51): icmp_seq=11. time=2. ms
64 bytes from server (11.0.0.51): icmp_seq=12. time=2. ms
ICMP Net Unreachable from gateway 17.0.0.1
 for icmp from client (16.0.0.51) to server (11.0.0.51)
ICMP Net Unreachable from gateway 17.0.0.1
 for icmp from client (16.0.0.51) to server (11.0.0.51)
64 bytes from server (11.0.0.51): icmp_seq=15. time=2. ms
64 bytes from server (11.
```

OSPF took approximately two seconds to detect and recover from the failed node.

The highlighted lines in following output from the `traceroute server` command shows the new path from the client to the server through the switch interface sw3.

```
client># traceroute server
traceroute: Warning: Multiple interfaces found; using 16.0.0.51 @
hme0
traceroute to server (11.0.0.51), 30 hops max, 40 byte packets
 1  16.0.0.1 (16.0.0.1)  0.699 ms  0.535 ms  0.581 ms
 2  15.0.0.1 (15.0.0.1)  1.481 ms  0.990 ms  0.986 ms
 3  14.0.0.1 (14.0.0.1)  1.214 ms  1.021 ms  1.002 ms
 4  13.0.0.1 (13.0.0.1)  1.322 ms  1.088 ms  1.100 ms
 5  12.0.0.1 (12.0.0.1)  1.245 ms  1.131 ms  1.220 ms
 6  server (11.0.0.51)  1.631 ms  1.200 ms  1.314 ms
```

The following code examples show the routing tables after the node failure. The first highlighted line in CODE EXAMPLE 5 shows the new route to the server through the switch sw3. The second highlighted line shows that the swithch sw2 link is down.

CODE EXAMPLE 5 Switch sw1 Routing Table After Node Failure

```
sw1:27 # sh ipr

 OR Destination        Gateway         Mtr   Flags     Use M-Use VLAN     Acct-1
*s  10.100.0.0/24      12.0.0.1          1 UG---S-um    63     0 net12    0
*oa 11.0.0.0/8         12.0.0.1          5 UG-----um   168     0 net12    0
*d  12.0.0.0/8         12.0.0.2          1 U------u-  1083     0 net12    0
*d  13.0.0.0/8         13.0.0.1          1 U------u-    41     0 net13    0
*oa 14.0.0.0/8         13.0.0.2          8 UG-----um     4     0 net13    0
*oa 15.0.0.0/8         13.0.0.2         12 UG-----um     0     0 net13    0
*oa 16.0.0.0/8         13.0.0.2         13 UG-----um    22     0 net13    0
*oa 17.0.0.0/8         13.0.0.2         12 UG-----um     0     0 net13    0
 d  18.0.0.0/8         18.0.0.1          1 ---------   515     0 -------- 0
*d  127.0.0.1/8        127.0.0.1         0 U-H----um     0     0 Default  0
```

CODE EXAMPLE 6 Switch sw2 Routing Table After Node Failure

```
sw1:4 # sh ipr

 OR Destination        Gateway         Mtr   Flags     Use M-Use VLAN     Acct-1
*s  10.100.0.0/24      12.0.0.1          1 UG---S-um    63     0 net12    0
*oa 11.0.0.0/8         12.0.0.1          5 UG-----um   168     0 net12    0
*d  12.0.0.0/8         12.0.0.2          1 U------u-  1102     0 net12    0
*d  13.0.0.0/8         13.0.0.1          1 U------u-    41     0 net13    0
```

CODE EXAMPLE 6 Switch sw2 Routing Table After Node Failure *(Continued)*

```
sw1:4 # sh ipr
*oa 14.0.0.0/8          13.0.0.2          8 UG-----um     4     0 net13    0
*oa 15.0.0.0/8          13.0.0.2         12 UG-----um     0     0 net13    0
*oa 16.0.0.0/8          13.0.0.2         13 UG-----um    22     0 net13    0
*oa 17.0.0.0/8          13.0.0.2         12 UG-----um     0     0 net13    0
 d  18.0.0.0/8          18.0.0.1          1 ---------   515     0 -------- 0
*d  127.0.0.1/8         127.0.0.1         0 U-H----um     0     0 Default  0
```

CODE EXAMPLE 7 Switch sw3 Routing Table After Node Failure

```
sw3:6 # sh ipr

 OR Destination        Gateway         Mtr   Flags     Use M-Use VLAN     Acct-1
*s  10.100.0.0/24      13.0.0.1          1 UG---S-um    26     0 net13    0
*oa 11.0.0.0/8         13.0.0.1          9 UG-----um    24     0 net13    0
*oa 12.0.0.0/8         13.0.0.1          8 UG-----um   134     0 net13    0
*d  13.0.0.0/8         13.0.0.2          1 U------u-    29     0 net13    0
*d  14.0.0.0/8         14.0.0.1          1 U------u-    20     0 net14    0
*oa 15.0.0.0/8         14.0.0.2          8 UG-----um     0     0 net14    0
*oa 16.0.0.0/8         14.0.0.2          9 UG-----um    25     0 net14    0
*oa 17.0.0.0/8         14.0.0.2          8 UG-----um     0     0 net14    0
*d  127.0.0.1/8        127.0.0.1         0 U-H----um     0     0 Default  0
```

The highlighted line in CODE EXAMPLE 8 shows the new route back to the client through sw3.

CODE EXAMPLE 8 Switch sw4 Routing Table After Node Failure

```
sw4:9 # sh ipr

 OR Destination        Gateway         Mtr   Flags     Use M-Use VLAN     Acct-1
*s  10.100.0.0/24      14.0.0.1          1 UG---S-um    29     0 net14    0
*oa 11.0.0.0/8         14.0.0.1         13 UG-----um    21     0 net14    0
*oa 12.0.0.0/8         14.0.0.1         12 UG-----um     0     0 net14    0
*oa 13.0.0.0/8         14.0.0.1          8 UG-----um     0     0 net14    0
*d  14.0.0.0/8         14.0.0.2          1 U------u-    12     0 net14    0
*d  15.0.0.0/8         15.0.0.1          1 U------u-   216     0 net15    0
*oa 16.0.0.0/8         15.0.0.2          5 UG-----um    70     0 net15    0
*d  17.0.0.0/8         17.0.0.2          1 U------u-    12     0 net17    0
*d  127.0.0.1/8        127.0.0.1         0 U-H----um     0     0 Default  0
```

OSPF is a good routing protocol with enterprise networks. It has fast failure detection and recovery. However, there are security concerns that should be investigated prior to deployment.

Layer 3—RIP Network Redundancy

The Routing Information Protocol (RIP) is based on the Bellman-Ford distance vector algorithm. The idea behind the RIP is that each RIP router builds a one-dimensional array that contains a scalar notion of hops, to reach all other hops. (OSPF was able to more accurately use the notion of cost, which could capture information such as link speed.) RIP routers flood each other with their view of the network by first starting with directly connected neighbor routers, then modifying their vector if they find from peer updates that there is a shorter path.

After a few updates, a complete routing table is constructed. When a router detects a failure, the distance is updated to infinity. Ideally, all routers would eventually receive the proper update and adjust their tables accordingly. However, if the network is designed with redundancy, there can be issues in properly updating the tables to reflect a failed link. There are problems such as count to infinity and split horizon, which have various fixes to this protocol. The RIP was a first implementation of the distance vector algorithm. The RIPv2, the most common, which addresses scalability and other limitations of the RIP.

To better understand the failover capabilities of RIPv2, the following test network was set up, shown in FIGURE 9.

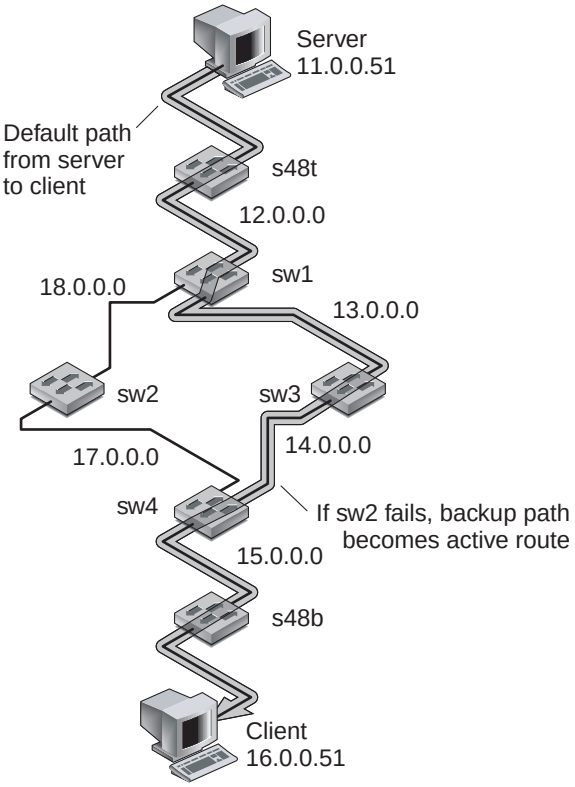

FIGURE 9 RIP Network Setup

CODE EXAMPLE 9 shows the server-to-client path before node failure. The highlighted lines show the path through the switch sw3.

CODE EXAMPLE 9 Server-to-Client Path Before Node Failure

```
server># traceroute client
traceroute: Warning: Multiple interfaces found; using 11.0.0.51
@ hme0
traceroute to client (16.0.0.51), 30 hops max, 40 byte packets
 1  11.0.0.1 (11.0.0.1)  0.711 ms  0.524 ms  0.507 ms
 2  12.0.0.2 (12.0.0.2)  1.448 ms  0.919 ms  0.875 ms
 3  13.0.0.2 (13.0.0.2)  1.304 ms  0.977 ms  0.964 ms
 4  14.0.0.2 (14.0.0.2)  1.963 ms  1.091 ms  1.151 ms
 5  15.0.0.2 (15.0.0.2)  1.158 ms  1.059 ms  1.037 ms
 6  client (16.0.0.51)  1.560 ms  1.170 ms  1.107 ms
```

The following code examples show the initial routing tables. The highlighted line in CODE EXAMPLE 10 shows the path to the client through the switch sw3.

CODE EXAMPLE 10 Switch sw1 Initial Routing Table

```
 OR Destination        Gateway         Mtr   Flags     Use M-Use VLAN     Acct-1
*s  10.100.0.0/24      12.0.0.1          1 UG---S-um    32     0 net12    0
*r  11.0.0.0/8         12.0.0.1          2 UG-----um    15     0 net12    0
*d  12.0.0.0/8         12.0.0.2          1 U------u-   184     0 net12    0
*d  13.0.0.0/8         13.0.0.1          1 U------u-    52     0 net13    0
*r  14.0.0.0/8         13.0.0.2          2 UG-----um     1     0 net13    0
*r  15.0.0.0/8         18.0.0.2          3 UG-----um     0     0 net18    0
*r  16.0.0.0/8         13.0.0.2          4 UG-----um    10     0 net13    0
*r  17.0.0.0/8         18.0.0.2          2 UG-----um     0     0 net18    0
*d  18.0.0.0/8         18.0.0.1          1 U------u-    12     0 net18    0
*d  127.0.0.1/8        127.0.0.1         0 U-H----um     0     0 Default  0
```

CODE EXAMPLE 11 Switch sw2 Initial Routing Table

```
sw2:3 # sh ipr

 OR Destination        Gateway         Mtr   Flags     Use M-Use VLAN     Acct-1
*s  10.100.0.0/24      18.0.0.1          1 UG---S-um    81     0 net18    0
*r  11.0.0.0/8         18.0.0.1          3 UG-----um     9     0 net18    0
*r  12.0.0.0/8         18.0.0.1          2 UG-----um    44     0 net18    0
*r  13.0.0.0/8         18.0.0.1          2 UG-----um     0     0 net18    0
*r  14.0.0.0/8         17.0.0.2          2 UG-----um     0     0 net17    0
*r  15.0.0.0/8         17.0.0.2          2 UG-----um     0     0 net17    0
*r  16.0.0.0/8         17.0.0.2          3 UG-----um     3     0 net17    0
*d  17.0.0.0/8         17.0.0.1          1 U------u-    17     0 net17    0
*d  18.0.0.0/8          18.0.0.2          1 U------u-   478     0 net18    0
*d  127.0.0.1/8         127.0.0.1         0 U-H----um     0     0 Default  0
#
#
```

CODE EXAMPLE 12 Switch sw3 Initial Routing Table

```
sw3:3 # sh ipr

 OR Destination        Gateway         Mtr   Flags     Use M-Use VLAN     Acct-1
*s  10.100.0.0/24      13.0.0.1          1 UG---S-um    79     0 net13    0
*r  11.0.0.0/8         13.0.0.1          3 UG-----um     3     0 net13    0
```

CODE EXAMPLE 12 Switch sw3 Initial Routing Table

```
sw3:3 # sh ipr
*r  12.0.0.0/8         13.0.0.1          2 UG-----um    44     0 net13    0
*d  13.0.0.0/8         13.0.0.2          1 U------u-    85     0 net13    0
*d  14.0.0.0/8         14.0.0.1          1 U------u-    33     0 net14    0
*r  15.0.0.0/8         14.0.0.2          2 UG-----um     0     0 net14    0
*r  16.0.0.0/8         14.0.0.2          3 UG-----um    10     0 net14    0
*r  17.0.0.0/8         14.0.0.2          2 UG-----um     0     0 net14    0
*r  18.0.0.0/8         13.0.0.1          2 UG-----um     0     0 net13    0
*d  127.0.0.1/8        127.0.0.1         0 U-H----um     0     0 Default  0
```

The highlighted line in CODE EXAMPLE 13 shows the path to the server through the switch sw3.

CODE EXAMPLE 13 Switch sw4 Initial Routing Table

```
sw4:7 # sh ipr

 OR Destination        Gateway         Mtr   Flags     Use M-Use VLAN     Acct-1
*s  10.100.0.0/24      14.0.0.1          1 UG---S-um    29     0 net14    0
*r  11.0.0.0/8         14.0.0.1          4 UG-----um     9     0 net14
*r  12.0.0.0/8         14.0.0.1          3 UG-----um     0     0 net14 0
*r  13.0.0.0/8         14.0.0.1          2 UG-----um     0     0 net14    0
*d  14.0.0.0/8         14.0.0.2          1 U------u-    13     0 net14    0
*d  15.0.0.0/8         15.0.0.1          1 U------u-   310     0 net15    0
*r  16.0.0.0/8         15.0.0.2          2 UG-----um    16     0 net15    0
*d  17.0.0.0/8         17.0.0.2          1 U------u-     3     0 net17    0
*r  18.0.0.0/8         17.0.0.1          2 UG-----um     0     0 net17    0
*d  127.0.0.1/8        127.0.0.1         0 U-H----um     0     0 Default  0
```

The highlighted lines in the following output from running the `traceroute client` command show the new path from the server to the client through the switch sw2 after the switch sw3 fails.

```
server># traceroute client
traceroute: Warning: Multiple interfaces found; using 11.0.0.51 @
hme0
traceroute to client (16.0.0.51), 30 hops max, 40 byte packets
 1  11.0.0.1 (11.0.0.1)  0.678 ms  0.479 ms  0.465 ms
 2  12.0.0.2 (12.0.0.2)  1.331 ms  0.899 ms  0.833 ms
 3  18.0.0.2 (18.0.0.2)  1.183 ms  0.966 ms  0.953 ms
 4  17.0.0.2 (17.0.0.2)  1.379 ms  1.082 ms  1.062 ms
 5  15.0.0.2 (15.0.0.2)  1.101 ms  1.024 ms  0.993 ms
 6  client (16.0.0.51)  1.209 ms  1.086 ms  1.074 ms
```

The following output shows the server `pings`.

```
64 bytes from client (16.0.0.51): icmp_seq=18. time=2. ms
64 bytes from client (16.0.0.51): icmp_seq=19. time=2. ms
64 bytes from client (16.0.0.51): icmp_seq=20. time=2. ms
ICMP Net Unreachable from gateway 12.0.0.2
 for icmp from server (11.0.0.51) to client (16.0.0.51)
ICMP Net Unreachable from gateway 12.0.0.2
..
..
for icmp from server (11.0.0.51) to client (16.0.0.51)
ICMP Net Unreachable from gateway 12.0.0.2
 for icmp from server (11.0.0.51) to client (16.0.0.51)
ICMP Net Unreachable from gateway 12.0.0.2
 for icmp from server (11.0.0.51) to client (16.0.0.51)
64 bytes from client (16.0.0.51): icmp_seq=41. time=2. ms
64 bytes from client (16.0.0.51): icmp_seq=42. time=2. ms
64 bytes from client (16.0.0.51): icmp_seq=43. time=2. ms
```

The fault detection and recovery took in excess of 21 seconds. The RIPv2 is widely available. However, the failure detection and recovery is not optimal. Further, there are other inherent robustness issues with the RIP.

Conclusion

We presented several approaches for increased availability for network designs by evaluating fault detection and recovery times and the adverse impact on computing and memory resources. In comparing these networking designs, we drew the following conclusions:

- Link aggregaton is suitable for increasing the bandwidth capacity and availability on point-to-point links only.
- Spanning Tree Protocol is not suitable because failure detection and recovery are slow. A recent improvement, IEEE 802.3w, Rapid Spanning Tree, designed to improve these limitations may be worth considering in the future
- The VRRP and IPMP is one of the most popular availability strategy combinations for server-to-network connection and a recommended approach. This approach provides rapid failure detection and recovery, is scalable, and economically feasible when considering the increased MTBF calculations.
- OSPF provides a much better failure detection and recovery than RIPv2 and is recommended for interswitch availablity. However, OSPF is not recommended to run a routing protocol on a server because of potential security issues, scalability,

and performance. The calculation of route tables from Link State Database in OSPF, for example, can impact server performance, depending on the number of routes in the autonomous system; the rate of changes in routing state; and the size of the server.

About the Authors

Deepak Kakadia (deepak.kakadia@eng.sun.com) is a staff engineer, network architect at Sun Microsystems Inc. Menlo Park, California. He has been with Sun for seven years, and has worked previously with Corona Networks as principal engineer in the Network Management Systems group, Digital Equipment Corp, where he worked on DEC OSF/1, and Nortel Networks (Bell Northern Research) in Ottawa, Canada. He received a Bachelor of Engineering in Computer Systems, a Master of Science in Computer Science, and has completed Ph.D. qualifiers and course work in Computer Science. He has filed four patents in the area of Network and Systems Management.

Sam Halabi is vice president of Marketing and Business Development at Extreme Networks in Sunnyvale, California. He is a seasoned executive and an industry veteran with over 15 years of experience in the development, sales, and marketing of complex data networking equipment to the worldwide enterprise and service provider markets. As executive vice president of Marketing and Business Development at Extreme Networks, Mr. Halabi was instrumental in creating Extreme's fast growing Ethernet Metro business and marketing its applications to large enterprise and service provider customers. Prior to Extreme, Mr. Halabi was vice president of Marketing at Pluris, a start-up targeting the Internet core routing market. Mr. Halabi held multiple leadership positions at Cisco Systems' high-end routing and switching divisions in the fields of marketing and network engineering. While at Cisco, Mr. Halabi wrote the first Cisco Internet routing book, a bestseller in the U.S and international markets. Mr. Halabi, started his data networking career at 3Com (1988-1993) in the field of engineering. Mr. Halabi served on the board of directors of the Optical Internetworking Forum (OIF) and the MPLS Forum. He holds a Master of Science in Computer Science from San Jose State University.

Bill Cormier(bcormier@extremenetworks.com) is a network systems engineer with Extreme Networks. Bill has worked for networking product companies like Extreme Networks, 3Com Corporation, and Racal-Datacom for the past 17 years, helping many companies design and operate highly available networks.

Ordering Sun Documents

The SunDocs℠ program provides more than 250 manuals from Sun Microsystems, Inc. If you live in the United States, Canada, Europe, or Japan, you can purchase documentation sets or individual manuals through this program.

Accessing Sun Documentation Online

The `docs.sun.com`℠ web site enables you to access Sun technical documentation online. You can browse the `docs.sun.com` archive or search for a specific book title or subject. The URL is `http://docs.sun.com/`

To reference Sun BluePrints™ OnLine articles, visit the Sun BluePrints OnLine web site at: `http://www.sun.com/blueprints/online.html`