Improve the Convergence of Mission-Critical Networks with Bidirectional Forwarding Detection (BFD)

By Ivan Pepelnjak

1. 11. 2008

A few years ago, we measured network convergence times in seconds, and aimed to establish an alternate path across the network following a link or node failure within a few seconds. Ten seconds was a sensible number that wouldn't upset users (assuming that the failures weren't too common) and definitely wouldn't break TCP sessions. (The DLSw local termination would take care of the leftovers of the SNA networks.) The only environments aiming at sub-second convergence times were real-time mission-critical applications, from industrial control to battlefield communications.

The introduction of Voice over IP (VoIP) into the data networks changed the rules completely; the human ear tends to be upset by even a sub-second loss of signal. While the quality-of-service requirements of VoIP deployment are usually well understood (and often implemented quite successfully), the convergence aspects are often forgotten—that is, until the first major failure, when even the proverbial phone doesn't ring any more.

Until recently, we've had to struggle with a patchwork of (frequently proprietary) kludges implemented on top of unreliable data-link layers (for example, frame relay end-to-end keepalives) and routing protocol "solutions" that offered only shortened *hello* timers. Fortunately, after struggling for more than 25 years with the deficiencies of Layer 2 signaling, major router vendors introduced an interoperable standard protocol aimed exclusively at detecting two-way communication with the next-hop router: Bidirectional Forwarding Detection (BFD).

Introduction to BFD

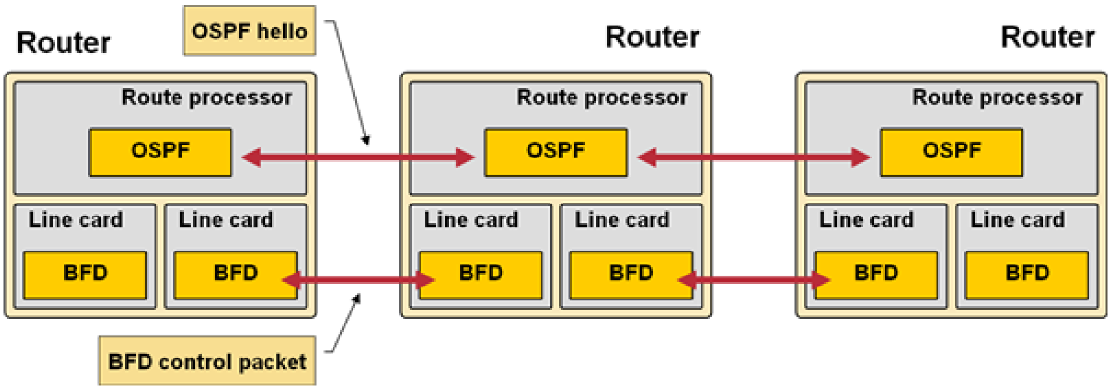
The Bidirectional Forwarding Detection protocol has a single, well-defined goal: "[P]rovide low-overhead, short-duration detection of failures in the path between adjacent forwarding engines, including the interfaces, data links and (to the extent possible) the forwarding engines themselves" (source: BFD draft specifications).

Note: Forwarding engines are commonly known as routers. The distinction between a router and a forwarding engine is significant only in distributed architectures with physically separate control and forwarding plane; for example, the Gigabit Switch Router (GSR).

The BFD protocol is very similar to the *hello* mechanisms used in the majority of the routing protocols, with a few significant differences:

- BFD tests bidirectional communication. (The EIGRP *hello* protocol can have problems with unidirectional links.)
- BFD packets are small (24 bytes on top of the UDP+IP header) and focused exclusively on path-failure detection. (Routing protocol *hello* packets perform a number of additional duties.)
- In routers with a distributed architecture, BFD packets can be processed on interface modules (see Figure 1), whereas routing protocol *hello* packets are always processed by the control plane (main CPU).

Figure 1: Distributed platforms run BFD on the line cards

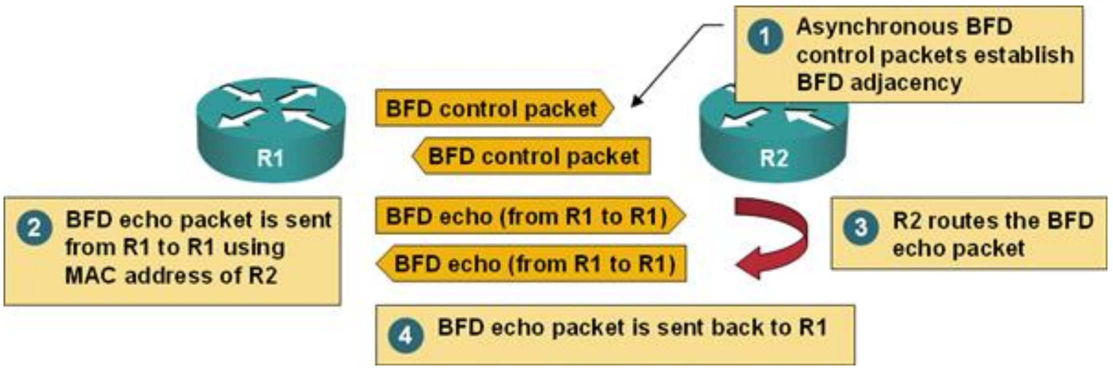


Note: Due to the low overhead, it's common to use BFD timers in the milliseconds range. (The minimum value accepted by Cisco IOS is 50 milliseconds.)

BFD can rely on control packets (similar to the *hello* packets used by the routing protocols), or use echo packets (IP packets addressed to the router itself but sent to the Layer 2 address of the next-hop node, as shown in Figure 2) in parallel with control packets. The echo packets thoroughly test the complete bidirectional forwarding path between adjacent routers, as they have to be transmitted by the originating router, propagated to the adjacent router, received by its interface logic, switched by its forwarding engine and sent back to the originator (similar to the way in which GRE keepalives work).

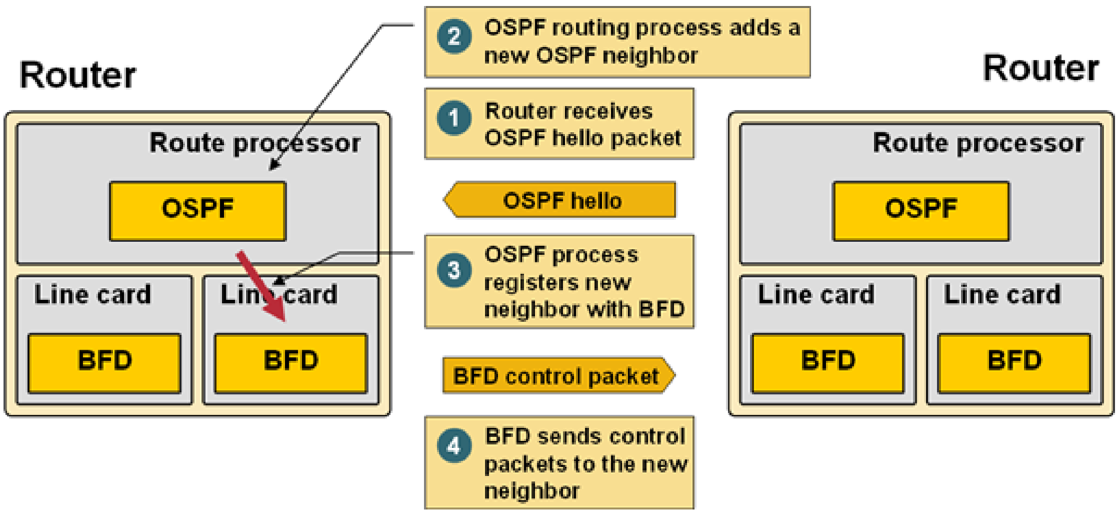
For example, when R1 sends a BFD echo packet in Figure 2, it sets the destination IP address in the packet to its own interface IP address and the MAC address in the Layer 2 frame header to the neighbor's (R2) MAC address. When R2 receives the packet, it performs a Layer 3 lookup and sends the packet toward its final destination (back to R1).

Figure 2: BFD echo mode


1 Asynchronous BFD control packets establish BFD adjacency
R1
BFD control packet
BFD control packet
R2
2 BFD echo packet is sent from R1 to R1 using MAC address of R2
BFD echo (from R1 to R1)
BFD echo (from R1 to R1)
3 R2 routes the BFD echo packet
4 BFD echo packet is sent back to R1

The BFD protocol has no inherent discovery mechanisms to detect adjacent BFD nodes; it's designed solely as an agent for other applications requiring fast failure-detection. Whenever a routing protocol (or any other application) configured to use BFD detects a new neighbor, it requests availability tracking from the BFD module, as shown in Figure 3. After the BFD session is established with the new neighbor, BFD signals adjacent node failure to the routing protocol, significantly reducing failure-detection time.

Figure 3: OSPF registers its neighbors with BFD


Router
Route processor
OSPF
Line card
Line card
BFD
BFD
2 OSPF routing process adds a new OSPF neighbor
1 Router receives OSPF hello packet
OSPF hello
3 OSPF process registers new neighbor with BFD
BFD control packet
4 BFD sends control packets to the new neighbor
Router
Route processor
OSPF
Line card
Line card
BFD
BFD

Using BFD in Cisco IOS

As of October 2008, the implementation of BFD on Cisco routers is still somewhat sporadic and depends on the hardware platform you're using and the software that the platform is running. All modern routing protocols (OSPF, IS-IS, EIGRP and BGP), as well as Hot Standby Router Protocol (HSRP), work with BFD. Some platforms can also use BFD to track the status of static routes.

The hardware support is sketchier: high-end routers (including ASR) running specialized IOS releases (for example, the 12.2 SRC release) support almost anything you might need in the core of your network. Integrated Service Routers (1800 through 3800) support BFD on Ethernet interfaces, but not on serial links. Older low-end routers don't support BFD at all.

Warning: If you want to run BFD on ISRs, use the latest maintenance build of the 12.4(15)T release or a later IOS release.

Configuring BFD is a two-step process:

- Step 1: Configure BFD timers on the interfaces on which you want to use BFD.
- Step 2: Configure routing protocols to use BFD.

Configuring BFD

BFD timers are configured with the **bfd interval *send-timer* mix_rx *receive-timer* multiplier *number*** interface configuration command. The *send-timer* specifies the frequency of BFD packets originated by the router, the *receive-timer* the minimum interval between packets accepted from BFD peers. (BFD adjacency will not form if the *send-timer* on one peer is lower than the *receive-timer* on another peer.) The multiplier *number* is the number of BFD packets that can be lost before the BFD peer is declared "down."

If you want to use the BFD echo mode, you should configure **bfd slow-timers** to specify the interval at which the control packets are sent and **bfd echo** on the interface to enable BFD echo mode.

Listing 1 shows a typical BFD configuration with BFD echo mode.

Listing 1: Typical BFD configuration

```
bfd slow-timers 3000

!

interface FastEthernet0/0

 bfd interval 100 min_rx 100 multiplier 3

 bfd echo
```

Configuring Interior Routing Protocols

After you've configured BFD on individual interfaces, you have to tell the routing protocols to use it. In most cases, there's no good reason that the routing protocols would *not* use BFD whenever possible; the only command you have to use in these scenarios is the **bfd all-interfaces** router configuration command. For example, to tell OSPF to use BFD as configured in Listing 1, use the OSPF configuration in Listing 2.

Listing 2: OSPF uses BFD on all interfaces

```
router ospf 1

 network 0.0.0.0 255.255.255.255 area 0

 bfd all-interfaces
```

If you want to be more specific, you could enable BFD on individual interfaces with the specific command for that particular routing protocol (for OSPF, use the **ip ospf bfd** interface configuration command), as shown in Listing 3.

Listing 3: OSPF uses BFD on Fast Ethernet interface only

```
interface FastEthernet0/0

 ip address 10.0.0.5 255.255.255.0

 ip ospf bfd

 bfd interval 100 min_rx 100 multiplier 3
```

You could also enable BFD on all interfaces with the **bfd all-interfaces** router configuration command, and disable it on specific interfaces. (OSPF uses the **ip ospf bfd disable** interface configuration command.)

Note: You'd disable BFD on an interface only if the interface is flaky but you want to retain routing adjacency across short failures. This shouldn't be necessary in most well-designed networks.

Using BFD with BGP

BGP can use BFD to detect failure of directly connected neighbors. eBGP or iBGP neighbor failure is detected as long as the neighbor resides on a directly connected subnet. The per-neighbor loss detection is configured with the **neighbor fall-over bfd** router configuration command. The configuration in Listing 4 uses BFD to detect failures of two BGP neighbors. One is directly connected (even though it uses an iBGP session); the other uses the more common loopback-to-loopback iBGP session.

Listing 4: Using BFD with BGP

```
interface FastEthernet0/0

 ip address 10.0.0.5 255.255.255.0

 bfd interval 100 min_rx 100 multiplier 3

!

router bgp 65000

 no synchronization

 bgp log-neighbor-changes

 neighbor 10.0.0.6 remote-as 65000

 neighbor 10.0.0.6 fall-over bfd

 neighbor 172.16.0.12 remote-as 65000

 neighbor 172.16.0.12 update-source Loopback0

 neighbor 172.16.0.12 fall-over bfd

 no auto-summary
```

When the router succeeds in establishing a BGP session with a neighbor configured with BFD failure detection, it tries to register the neighbor's IP address with BFD. Directly connected neighbors are accepted by BFD. Destinations of multihop BGP sessions are obviously rejected (BFD for multihop paths is not yet supported by Cisco IOS), resulting in a *syslog* message, as shown in Listing 5.

Listing 5: BFD does not work for multihop BGP neighbors

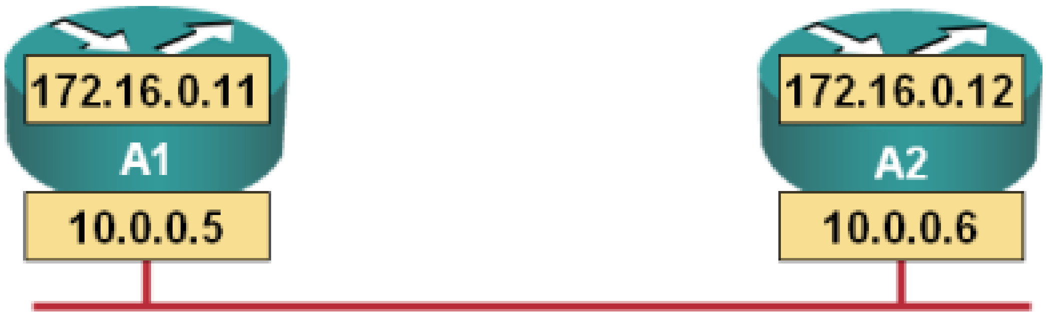
```
%BGP-4-
BFD_NOT_ONEHOP: BFD is supported for single hop neighbors. 172.16.0.12 is no
t single
hop neighbor

%BGP-5-ADJCHANGE: neighbor 172.16.0.12 Up
```

BFD in Action

n Figure 4, I've set up a very simple lab network to illustrate BFD failure-detection capabilities: two OSPF routers connected over a Fast Ethernet link. I've synchronized the times with Network Time Protocol to ensure that the *syslog* messages will indicate proper chronological order of events.

Figure 4: BFD test network



In this example, I've configured BFD on the Fast Ethernet interfaces (remember that BFD doesn't work on serial interfaces yet on the 2800 series routers) and configured the OSPF routing protocol to use it with the configuration commands from Listing 3. After configuring BFD and OSPF, I've used the **show bfd neighbor** command to check the BFD neighbors (Listing 6).

Listing 6: BFD neighbors registered by OSPF

```
a1#show ip ospf neighbor
Neighbor ID   Pri  State      Dead Time Address    Interface
172.16.0.21     0  FULL/  -   00:00:37  172.16.1.2 Serial0/0/0.100
172.16.0.12     1  FULL/BDR   00:00:31  10.0.0.6   FastEthernet0/0
a1#show bfd neighbor
OurAddr    NeighAddr  LD/RD  RH/RS  Holddown(mult)  State  Int
10.0.0.5   10.0.0.6    1/2    Up         0    (3 )  Up     Fa0/0
```

If you want to see which protocols use BFD to detect failure of a particular BFD neighbor, use the **show bfd neighbors details** command. The printout in Listing 7 indicates that both OSPF and HSRP use BFD to improve neighbor failure-detection times.

Listing 7: BFD is used by OSPF and HSRP

```
a1#show bfd neighbors details
OurAddr    NeighAddr  LD/RD  RH/RS  Holddown(mult)  State  Int
10.0.0.5   10.0.0.6    1/2    Up         0    (3 )  Up     Fa0/0
Session state is UP and using echo function with 100 ms interval.
Local Diag: 0, Demand mode: 0, Poll bit: 0
MinTxInt: 1000000, MinRxInt: 1000000, Multiplier: 3
Received MinRxInt: 1000000, Received Multiplier: 3
```

```
Holddown (hits): 0(0), Hello (hits): 1000(866)

Rx Count: 525, Rx Interval (ms) min/max/avg: 1/23596/923 last: 176 ms ago

Tx Count: 869, Tx Interval (ms) min/max/avg: 1/1000/869 last: 624 ms ago

Elapsed time watermarks: -1 0 (last: 0)

Registered protocols: OSPF HSRP

Uptime: 00:05:59

Last packet: Version: 1             - Diagnostic: 0

             State bit: Up          - Demand bit: 0

             Poll bit: 0            - Final bit: 0

             Multiplier: 3          - Length: 24

             My Discr.: 2           - Your Discr.: 1

             Min tx interval: 1000000     - Min rx interval: 1000000

             Min Echo interval: 100000
```

Failure Detection with BFD

Without BFD, neighbor failure-detection relies exclusively on OSPF *hello* timers. A1 thus realized that A2 was no longer available approximately half a minute after I shut down the Fast Ethernet interface on A2 (Listing 8).

Listing 8: Neighbor failure-detection with OSPF

```
a2(config)#int fa 0/0

a2(config-if)#shut

a2(config-if)#

17:31:27.347: %LINK-5-
CHANGED: Interface FastEthernet0/0, changed state to administratively down

a1#

17:32:01.723: %OSPF-5-
ADJCHG: Process 1, Nbr 172.16.0.12 on FastEthernet0/0 from FULL to DOWN, Nei
ghbor
Down: Dead timer expired
```

When using BFD, A1 detected that A2 was no longer reachable long before A2 had time to generate the *syslog* message indicating that the interface had been disabled (Listing 9).

Listing 9: OSPF neighbor loss-detection with BFD

```
a2(config-if)#shut

a2(config-if)#

17:34:10.304: %OSPF-5-
ADJCHG: Process 1, Nbr 172.16.0.11 on FastEthernet0/0 from FULL to DOWN, Nei
ghbor
Down: Interface down or detached

17:34:12.300: %LINK-5-
CHANGED: Interface FastEthernet0/0, changed state to administratively down

a1#

17:34:10.615: %OSPF-5-
ADJCHG: Process 1, Nbr 172.16.0.12 on FastEthernet0/0 from FULL to DOWN, Nei
ghbor
Down: BFD node down
```

Summary

In this article, you've seen how the introduction of Bidirectional Forwarding Detection (BFD) can significantly reduce network convergence time in environments that don't provide fast link or node failure-detection. BFD enables you to detect hop-by-hop Layer 3 failures in milliseconds without having to fine-tune routing protocol *hello* timers. The only changes required in the router configuration are the configuration of per-interface BFD parameters with the **bfd interval** configuration command and the enabling of BFD functionality in the routing protocols with the **bfd all-interfaces** router configuration command (or a protocol-specific configuration command, if you want to configure BFD only on particular interfaces).

NIL – More Than Just a Training Company

NIL Learning delivers the leading-edge Cisco training to IT professionals and companies around the globe. Through field-proven experts — each both active engineer and instructor — NIL Learning enhances the standard learning curriculum with real-life experience and helps clients to maximize their training investment.

NIL Learning is part of NIL, a leading global IT solutions provider. Since 1992, NIL has been at the forefront of advanced contributors to strategic partner Cisco's technologies, learning curriculum and value-added solutions deployed to clients around the globe. Today, NIL has earned the highest certifications offered by Cisco, VMware, EMC, HP, IBM, Microsoft, F5, Jive, MobileIron, RSA, VCE and others. Their portfolio of solutions consists of managed services, professional services and learning services.

NIL is headquartered in Slovenia, with regional offices in Croatia, Serbia, Saudi Arabia, the U.S., Turkey, South Africa, Morocco, Nigeria, Kenya and Botswana.

Why learn at NIL LEARNING?

- All NIL LEARNING instructors are **field-proven experts** - each both active engineer and instructor.
- **75% of NIL LEARNING engineers hold CCSI** certifica- tions, and **18 have already achieved the respected CCIE rank**.
- NIL LEARNING **enhances the standard learning curriculum** with real-life experience and helps clients to maximize their training investment.
- NIL has been a Cisco Training Partner for many years; it became a **Cisco Learning Partner in 1993**, and has been a **Cisco Gold Partner since 1995**.
- NIL was awarded the **Cisco Most Business Relevant Learning Partner in MEA in 2010 and the most innova- tive learning partner in MEA**.

- NIL received the **Innovation Award for its Technology Led Training and its extensive contribution to Cisco learning solutions** at the Cisco EMEAR Learning Partner Summit in 2012.
- NIL received the **Innovation Award for its Technology Led Training and Advanced Engineer Program** at the Cisco Global Learning Partner Summit in 2013.
- NIL LEARNING runs a **centralized training schedule across the whole EMEAR region**.

More Info

Slovenia

T: +386 1 4746 500

E: sales-support@nil.com

Saudi Arabia

T: +966 1 465 4641

E: info.nilme@nil.com

Botswana

T: +267 318 1684

E: training@it-iq.bw

Serbia

T: +381 11 2282 818

E: info-nilserbia@nil.co.rs

Croatia

T: +385 (0)51 583 255

E: info-nilcroatia@nil.com

South Africa

T: +27 (0)11 575 4637

E: mea_sales@nil.com

Kenya

T: +27 (0)11 575 4637

E: mea_sales@nil.com

Turkey

T: +902 123 81 8639

E: info-nilturkey@nil.com

Morocco

T: +212(0) 660 808 394

E: info-nilmorocco@nil.com

USA

T: +1 612 886 3900

E: info-nilusa@nil.com

Nigeria

T: +27 (0)11 575 4637

E: mea_sales@nil.com

www.learning.nil.com