

# A Study of the BGP Routing Protocol

**September 22, 2004**  
**Prepared for: Felix Carapaica**  
**Prepared by: John Tewfik**  
**Leif Madsen**  
**Drew Farlinger**

## Introduction

The purpose of this lab is to investigate the mechanisms and characteristics of the BGP-4 routing protocol. BGP is an exterior gateway protocol fashioned on distant vector principles. In our lab to effectively investigate BGP, considerations for peering, the usage of interior gateway protocol and layer two connectivity had to be considered.

Our investigation will be broken up into four parts;

- BGP dialogue
- Options, Flags and Path Attributes
- Routing Information
- Finite State Machine (FSM)

The BGP dialogue section will discuss the conversations between BGP peers. A breakdown of dialogue on a field by field basis as observed with our network analyzer Ethereal. With this we can gain an understanding of how these routers communicate, propagate information and build routing knowledge.

We will then discuss the Options, Flags and Path Attributes associated with the BGP protocol. As part of the BGP basics, three path attributes will be explored, including Origin, AS\_Path and Next\_Hop. We will focus chiefly on these attributes and their significance in routing decisions. The options and flags fields will also be discussed.

The routing tables in our exterior and interior protocols will be compared and discussed. What key elements of the EGP's routing table are evident and what problems we encountered.

Finite State Machine will be explored and dissected as this mechanism is the driving force behind the logic and decision making process of the BGP protocol.

## Procedure

Our lab starts with the creation of several autonomous systems. These autonomous systems are defined physically by the use of 3 computers (routers) at each table in the lab. Each router at the table will contain 2 network interfaces; one for a connection to the interior domain and another connected to the exterior domain.

These 10 autonomous systems will all be linked together through peering arrangements on each computer within the autonomous systems, connecting to one computer in another autonomous system. For example, a peering arrangement will be established between Host 1 of table 1, and Host 1 of table 7.

In the end we will have a partially-meshed topology between all of the autonomous systems, allowing us to fully investigate the ways that BGP peers talk to each other, and what we see inside of the messages sent as part of this communication dialog.

We will continue in detail the specifics of the construction of our network at each layer applicable to this lab.

## Layer 2

Let's take a look at how our autonomous systems will communicate at the layer 2 level. The lab is composed of a switch for each table. Each one of these switches will be partitioned into 3 separate VLAN's, to each of which one host at each table will belong to. For example, VLAN 12 will be created, and the first host from each autonomous system will be on this VLAN. Likewise, the other VLAN's – 22 and 32 – will be allocated to the 2<sup>nd</sup> and 3<sup>rd</sup> hosts of each autonomous system. The default VLAN for each autonomous system (table) will include the primary NIC of each host.

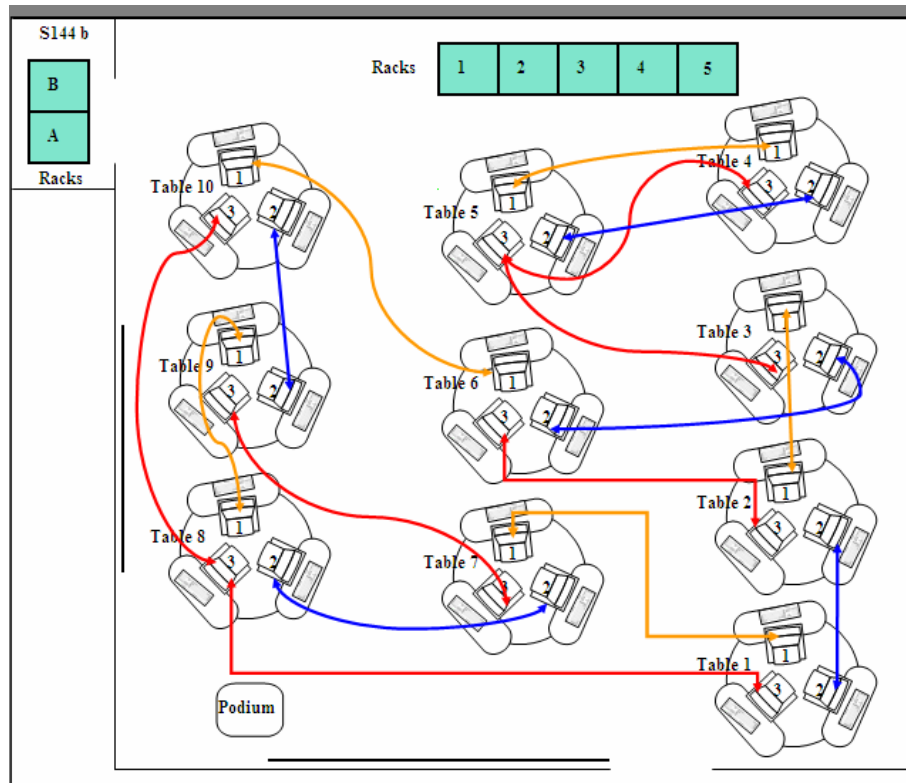


Figure 1-1: Peering Arrangement<sup>1</sup>

## Layer 3

Each router of the autonomous system will be running the zebra routing software. This will include the BGP and OSPF daemons. The addressing schema for each VLAN is outlined below. After successfully creating the OSPF domain which is limited to our AS, BGP peering can be implemented.

- Define VLAN12 (VLAN for all desktops 1, second port) for ports 5, and 6 of each switch.
- Define VLAN22 (VLAN for all desktops 2, second port) for ports 7, and 8 of each switch.
- Define VLAN32 (VLAN for all desktops 3, second port) for ports 9, and 10 of each switch.
- Assign IP address range 172.16.12.0 / 24 to VLAN12 of each switch.

<sup>1</sup> Image taken from Lab 1 PowerPoint presentation, provided by Felix Carapaica

- Assign IP address range 172.16.22.0 / 24 to VLAN22 of each switch.
- Assign IP address range 172.16.32.0 / 24 to VLAN32 of each switch.
- Assign IP address 172.16.VLAN#.table# / 24 to the interface connected.
- Extend the VLANs to all switches by using the optical fibre-GBICs connected to the 3550 multilayer switch<sup>2</sup>

### *Router Configuration*

We have included a sample configuration for the first router in our autonomous system. This will be broken down into the internal and external network configurations for the OSPF and BGP daemons respectively. The configurations will be similar for all routers in the lab environment.

#### ospfd.conf

```
router ospf
  ospf router-id 192.168.1.1
  redistribute bgp
  network 192.168.1.0/24 area 0.0.0.0
```

Fundamentals of the OSPF architecture were explored last term; thusly only key command lines are of interest. Of particular importance is the `redistribute bgp` command. This is used to *redistribute* the BGP routing information into the OSPF domain.

#### bgpd.conf

```
router bgp 65001
  bgp router-id 1.1.1.1
  network 192.168.1.0/24
  redistribute ospf
  neighbour 172.16.12.7 remote-as 65007
```

A line by line breakdown of this configuration is necessary to appreciate the logistics of BGP peering. The line containing `router bgp 65001` defines the BGP AS number of our router. All lines contained under this heading are used for the configuration of this BGP speaker. The router-id of the BGP router is defined on the next line as 1.1.1.1. We advertise the reachable internal networks with the use of the `network` command. Network 192.168.1.0/24 is reachable via our BGP router, allowing the computer peering with us (and any other computers peered with them) to reach computers within this network. In order for this network layer reachability information to be circulated to the peers, we must `redistribute ospf` information to our peers. The final line of the router configuration defines the IP address and AS number of our BGP neighbour.

---

<sup>2</sup> Information taken from Lab 1 PowerPoint presentation, provided by Felix Carapaica

## BGP Dialogue

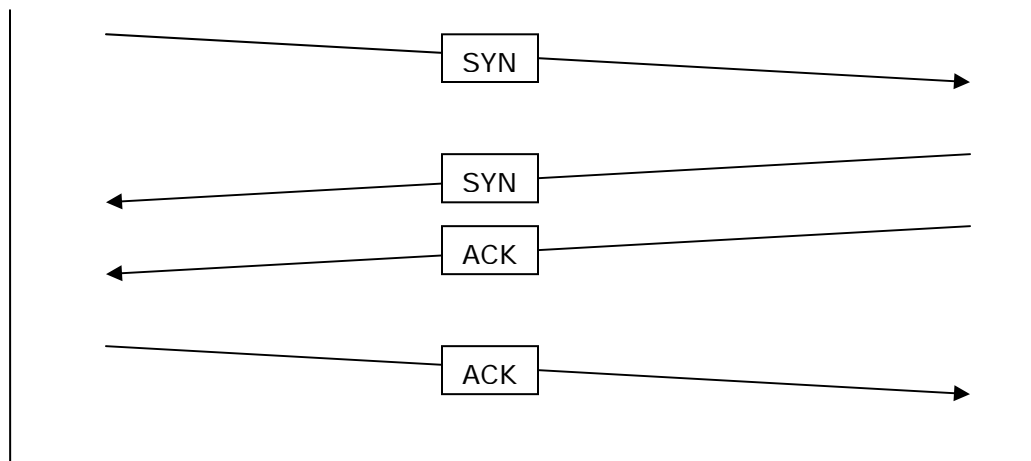
The messages between BGP speakers are of particular interest as they are the window into the methodology used by the BGP daemon to peer and distribute information. These messages are broken into four types:

- OPEN
- KEEPALIVE
- NOTIFICATION
- UPDATE

We will investigate each of these messages and provide an example taken from our network analyzer.

Before investigating these messages, some key issues must be addressed in terms of where this protocol is located in the OSI model and what mechanisms it makes use of at lower levels. Perhaps the most integral element of BGP communications is its usage of the transport layer and the TCP protocol. By taking advantage of all the tools available through the use of the TCP protocol, BGP exploits the portability of TCP across many types of environments. This reduces the amount of complexity within the BGP protocol, making it simpler and more modular. TCP is a connection-oriented service insuring delivery of messages via acknowledgements and retransmission. With this, BGP can assume messages are successfully received, without actually having to do any of the work.

As in all TCP communications, a 3 way handshake is necessary to kick off the BGP parlance. TCP port 179 is used to initially open the BGP session through the use of the TCP SYN, SYN ACK, ACK sequence. After this is complete, an OPEN message is sent to the remote peer. The OPEN message is then ACK'd by TCP and the same process is repeated in the opposite direction. This OPEN message includes important details of how these speakers are to communicate.



*Figure 2-1: The 3 Way TCP Handshake*

## The OPEN Message

The following is a fully expanded OPEN message as captured by Ethereal.

```

Border Gateway Protocol
  OPEN Message
    Marker: 16 bytes
    Length: 45 bytes
    Type: OPEN Message (1)
    Version: 4
    My AS: 65007
    Hold time: 180
    BGP identifier: 7.7.7.1
    optional parameters length: 16 bytes
  optional parameters
    Capabilities Advertisement (8 bytes)
      Parameter type: Capabilities (2)
      Parameter length: 6 bytes
      Multiprotocol extensions capability (6 bytes)
        Capability code: Multiprotocol extensions capability (1)
        Capability length: 4 bytes
        Capability value
          Address family identifier: IPv4 (1)
          Reserved: 1 byte
          Subsequent address family identifier: unicast (1)
    Capabilities Advertisement (4 bytes)
      Parameter type: Capabilities (2)
      Parameter length: 2 bytes
      Route refresh capability (2 bytes)
        Capability code: Route refresh capability (128)
        Capability length: 0 bytes
    Capabilities Advertisement (4 bytes)
      Parameter type: Capabilities (2)
      Parameter length: 2 bytes
      Route refresh capability (2 bytes)
        Capability code: Route refresh capability (2)
        Capability length: 0 bytes
```

Figure 2-2: OPEN message

The OPEN message is used to negotiate parameters needed for information exchange. Certain fields must be compatible with router configuration for information exchange to commence. Example: AS number must match the statically defined neighbour as found in the BGP configuration. If any part of this fails, a different BGP message, NOTIFICATION, is used to describe the error encountered. The notification message will be further discussed later in this paper.

The opening fields in this message will be explained in detail.

*Marker* – used for authentication purposes, but not implemented in this lab. The marker field is then full ones to indicate that no authentication is being used.

*Length* – indicated the length of the message in bytes.

*Type* – indicates the type of message (in this case, OPEN)

*Version* – the BGP version of the protocol being used.

*My AS* – this indicates the AS number of the source router and must match the statically defined neighbour command as found in the destination router.

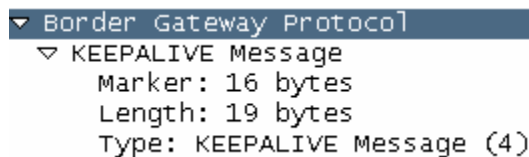
*Hold Time* – the maximum allowable time between routers where no KEEPALIVE or UPDATE messages are received. If the *Hold Time* is set to 0, then the timer never expires and the speaker is always considered available. The minimum *Hold Time* that can be configured is 3 seconds. For the purpose of our lab, we used the default 180 second value.

*BGP Identifier* – is the identifying address of the router in the format of a 4-octet IP address. If not statically defined with the use of the loopback address, the identifier will identify to the highest IP address available to the router.

*Optional Parameters Length* – this field along with the optional parameters itself dictate any superfluous settings that could be used to maximize and expand upon BGP communications. These options and parameters will be explained in the next section.

### *KEEPALIVE Message*

Below is a captured KEEPALIVE message from our network analyzer.



```
▼ Border Gateway Protocol
  ▼ KEEPALIVE Message
    Marker: 16 bytes
    Length: 19 bytes
    Type: KEEPALIVE Message (4)
```

*Figure 2-3: The KEEPALIVE Message*

As we see from the above diagram the common fields of a BGP message are present. Marker, Length and Type are evident as in the OPEN message, the only difference being Type 4 indicated a KEEPALIVE message.

The KEEPALIVE message is used to insure that speakers are aware of the remote end still being available. This message resets the Hold Timer field whenever the message is received. This has an affect on the Finite State Machine (FSM) logic which will be discussed later.

The KEEPALIVE message is only 19 bytes long which is the header length of all BGP messages. In addition to the KEEPALIVE message resetting the timer any update will perform the same task.

## NOTIFICATION Message

Figure 2-4 shows the expanded view of the NOTIFICATION message as captured by our network analyzer.

```
Border Gateway Protocol
  NOTIFICATION Message
    Marker: 16 bytes
    Length: 23 bytes
    Type: NOTIFICATION Message (3)
    Error code: OPEN Message Error (2)
    Error subcode: Bad Peer AS (2)
    Data
```

Figure 2-4: The NOTIFICATION Message

The BGP NOTIFICATION message is used to communicate errors between speakers during peer negotiation. Only key fields will be focused on as the header has already been described previously in this section.

*Error code* – Indicates at what point the error is happening in the peer negotiation.

*Error subcode* – used to indicate the specific error which occurred. In this case a faulty AS number was deliberately entered to invoke this error.

*Data* - The Data field returns data relevant to the specific error to further clarify the source of communication breakdown.

## UPDATE Message

Below you will find a screenshot of the UPDATE message as captured by our network analyzer. This is the life blood of the BGP routing protocol where integral information is exchanged between speakers, later used to make routing decisions.

```
Border Gateway Protocol
  UPDATE Message
    Marker: 16 bytes
    Length: 52 bytes
    Type: UPDATE Message (2)
    Unfeasible routes length: 0 bytes
    Total path attribute length: 25 bytes
    Path attributes
    Network layer reachability information: 4 bytes
    192.168.1.0/24
      NLRI prefix length: 24
      NLRI prefix: 192.168.1.0 (192.168.1.0)
```

Figure 2-5: The UPDATE Message

After the OPEN messages are negotiated by each speaker without error, the UPDATE messages are free to be delivered by each peer. Within the UPDATE message network information is propagated relevant to the speaking peer. This includes reachable networks seen by the speaking peer, routes not reachable and those that need to be withdrawn. In addition to this Path Attribute information is included which will be discussed later.

*Unfeasible routes length* – used to describe the length of the withdrawn routes in the message in bytes. The UPDATE message has a list of multiple routes or no routes at all which is indicated with a length of zero.

*Total Path Attribute Length* – is a length field in bytes of the path attributes that are to follow later in the message.

*Path Attributes* – refer to the specifics of how a network is to be reached and displays evidence of the topologies interconnection. This will be discussed in greater detail in the next section.

*Network Layer Reachability Information (NLRI)* – this portion of the update message indicated reachable networks using a length, prefix tuple. The length portion is the network mask length to be applied to the prefix, read from the most significant to least significant bit (left to right). The prefix is the network address portion and will be incorporated into the routing table.

## Options and Path Attributes

In this section we will discuss options and path attributes. The first, options, are detailed information propagated through the use of the OPEN message. These options include various capabilities which are an extension of the original protocol specification. Below is an expanded view of these extensions.

```
optional parameters length: 16 bytes
  ▾ Optional parameters
    ▾ Capabilities Advertisement (8 bytes)
      Parameter type: Capabilities (2)
      Parameter length: 6 bytes
      ▾ Multiprotocol extensions capability (6 bytes)
        Capability code: Multiprotocol extensions capability (1)
        Capability length: 4 bytes
        ▾ Capability value
          Address family identifier: IPv4 (1)
          Reserved: 1 byte
          Subsequent address family identifier: Unicast (1)
      ▾ Capabilities Advertisement (4 bytes)
        Parameter type: Capabilities (2)
        Parameter length: 2 bytes
        ▾ Route refresh capability (2 bytes)
          Capability code: Route refresh capability (128)
          Capability length: 0 bytes
      ▾ Capabilities Advertisement (4 bytes)
        Parameter type: Capabilities (2)
        Parameter length: 2 bytes
        ▾ Route refresh capability (2 bytes)
          Capability code: Route refresh capability (2)
          Capability length: 0 bytes
```

Figure 3-1: Optional Parameters

The first few fields of this section of the OPEN message are noted as parameter type and parameter length giving insight into the type of parameter and the information that is to follow. The multiprotocol extensions capability includes multiprotocol reachable and unreachable information. This information can be used for several purposes:

- to advertise a feasible route
- to advertise the network address of a next hop router
- to report subnetwork points

In addition to these multiprotocol extensions information regarding unicast or multicast capabilities is included.

Some flags in the UPDATE message should be examined to determine transitive and non-transitive information that is passed between routers.

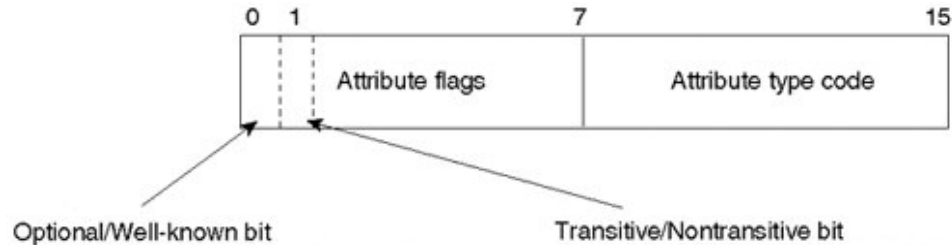


Figure 3-2: Attribute Flags<sup>3</sup>

There are four categories that path attributes fall under: well-known mandatory, well-known discretionary, optional transitive and optional non-transitive. These four categories are described by the first two bits of the attribute flags field. This flags field appears in every subsection of the path attributes contained within an UPDATE message.

The transitive flag is indicating information that must be passed between autonomous systems. Likewise, the reverse is true for attributes marked as non-transitive. An attribute which is well-known is one that all implementations of BGP will understand. It can be observed that some of these path attributes will not be understood by implementations that are not as fully compliant to RFC specification.

```

    ▾ Path attributes
      ▾ ORIGIN: IGP (4 bytes)
        ▸ Flags: 0x40 (well-known, Transitive, Complete)
          Type code: ORIGIN (1)
          Length: 1 byte
          origin: IGP (0)
      ▾ AS_PATH: 65001 (7 bytes)
        ▸ Flags: 0x40 (well-known, Transitive, Complete)
          Type code: AS_PATH (2)
          Length: 4 bytes
      ▾ AS path: 65001
        ▾ AS path segment: 65001
          Path segment type: AS_SEQUENCE (2)
          Path segment length: 1 AS
          Path segment value: 65001
      ▾ NEXT_HOP: 172.16.12.1 (7 bytes)
        ▸ Flags: 0x40 (well-known, Transitive, Complete)
          Type code: NEXT_HOP (3)
          Length: 4 bytes
          Next hop: 172.16.12.1 (172.16.12.1)
      ▾ MULTI_EXIT_DISC: 0 (7 bytes)
        ▸ Flags: 0x80 (Optional, Non-transitive, Complete)
          Type code: MULTI_EXIT_DISC (4)
          Length: 4 bytes
          Multiple exit discriminator: 0
      ▾ Network layer reachability information: 4 bytes
  
```

Figure 3-3: Path Attributes

<sup>3</sup> Image taken from Internet Routing Architectures, Second Edition by Sam Halabi, Chapter 5.

The above figure is a fully expanded view of the path attributes section in an UPDATE message. The subsections will be described below.

*ORIGIN* – this path attribute indicates the origin of the network information being communicated to a speaker. In our lab our IGP of choice was OSPF version 2. Although not explicitly stated in this attribute, the daemon is still aware that this information is being derived from an interior gateway protocol (IGP). The flags mark this portion as well-known, transitive, complete.

*AS\_PATH* – is the autonomous system path followed to reach given network prefix. As the message traverses through the BGP networks, each AS system appends its AS number to this path. This is used to prevent routing loops. In our lab AS\_PATH was a particular attribute of interest, one that raised questions which will be discussed in our final section. The flags for this attribute are marked as well-known, transitive, complete.

*NEXT\_HOP* – as its name sake this path attribute indicates the source network layer address for the use in building routing table knowledge. As the speaker forwards its message to a BGP peer, this field will indicate from who this message derives. This attribute is marked as a well-known, transitive, complete portion of the path attribute.

There are many path attributes which are not included in our implementation of BGP and for the purposes of this report, will not be examined.

## Finite State Machine (FSM)

We took a look at the BGP daemon Finite State Machine (FSM) to understand the behaviour of BGP under certain circumstances.

These observations were made when there was no activity on the network, and when there was some kind of change to the network, such as a disconnection. Also, when invalid peering information was configured.

First we'll take a look at the behaviour of the FSM when the network is idle.

```
2004/09/13 12:43:21 BGP: 172.16.22.2 [FSM] KeepAlive_timer_expired (Established->Established)
2004/09/13 12:43:21 BGP: 172.16.22.2 [FSM] Receive_KEEPA_LIVE_message (Established->Established)
2004/09/13 12:43:22 BGP: 172.16.22.2 [FSM] Timer (routeadv timer expire)
2004/09/13 12:43:52 BGP: 172.16.22.2 [FSM] Timer (routeadv timer expire)
2004/09/13 12:44:21 BGP: 172.16.22.2 [FSM] Timer (keepalive timer expire)
2004/09/13 12:44:21 BGP: 172.16.22.2 [FSM] KeepAlive_timer_expired (Established->Established)
2004/09/13 12:44:21 BGP: 172.16.22.2 [FSM] Receive_KEEPA_LIVE_message (Established->Established)
```

By taking a look at this excerpt from the log file, you can see that every minute the KEEPA\_LIVE timer will expire for any peer that you may have. As described earlier, this KEEPA\_LIVE timer can be adjusted to target the needs of the network. You will notice that as soon as the KEEPA\_LIVE timer expires, a message will be sent and received to keep the routers talking and aware of the fact that they are up. It can be observed that the BGP session between the routers has already been established by the (Established->Established) messages in the log file.

We researched the `Timer (routeadv timer expire)` message, and it may be that it is an error in the FSM coding of the Zebra implementation of BGP.

Next we'll take a look at the sequence of events that unfold when there is a change in the network. In this case we disconnected the cable from the interface of the router outputting to this log file.

```
2004/09/13 12:51:21 BGP: 172.16.22.2 [FSM] Receive_KEEPA_LIVE_message (Established->Established)
2004/09/13 12:51:22 BGP: 172.16.22.2 [FSM] Timer (routeadv timer expire)
2004/09/13 12:51:52 BGP: 172.16.22.2 [FSM] Timer (routeadv timer expire)
2004/09/13 12:52:21 BGP: 172.16.22.2 [FSM] Timer (keepalive timer expire)
2004/09/13 12:52:21 BGP: 172.16.22.2 [FSM] KeepAlive_timer_expired (Established->Established)
2004/09/13 12:52:22 BGP: 172.16.22.2 [FSM] Timer (routeadv timer expire)
2004/09/13 12:52:52 BGP: 172.16.22.2 [FSM] Timer (routeadv timer expire)
2004/09/13 12:53:21 BGP: 172.16.22.2 [FSM] Timer (keepalive timer expire)
2004/09/13 12:53:21 BGP: 172.16.22.2 [FSM] KeepAlive_timer_expired (Established->Established)
2004/09/13 12:53:22 BGP: 172.16.22.2 [FSM] Timer (routeadv timer expire)
2004/09/13 12:53:52 BGP: 172.16.22.2 [FSM] Timer (routeadv timer expire)
2004/09/13 12:54:21 BGP: 172.16.22.2 [FSM] Timer (holdtime timer expire)
2004/09/13 12:54:21 BGP: 172.16.22.2 [FSM] Hold_Timer_expired (Established->Idle)
2004/09/13 12:54:21 BGP: 172.16.22.2 [FSM] Hold timer expire
2004/09/13 12:54:31 BGP: 172.16.22.2 [FSM] Timer (start timer expire).
2004/09/13 12:54:31 BGP: 172.16.22.2 [FSM] BGP_Start (Idle->Connect)
2004/09/13 12:54:31 BGP: 172.16.22.2 [FSM] Non blocking connect waiting result
2004/09/13 12:54:34 BGP: 172.16.22.2 [FSM] TCP_connection_open_failed (Connect->Active)
2004/09/13 12:56:34 BGP: 172.16.22.2 [FSM] Timer (connect timer expire)
2004/09/13 12:56:34 BGP: 172.16.22.2 [FSM] ConnectRetry_timer_expired (Active->Connect)
2004/09/13 12:56:34 BGP: 172.16.22.2 [FSM] Non blocking connect waiting result
2004/09/13 12:56:37 BGP: 172.16.22.2 [FSM] TCP_connection_open_failed (Connect->Active)
2004/09/13 12:58:37 BGP: 172.16.22.2 [FSM] Timer (connect timer expire)
2004/09/13 12:58:37 BGP: 172.16.22.2 [FSM] ConnectRetry_timer_expired (Active->Connect)
2004/09/13 12:58:37 BGP: 172.16.22.2 [FSM] Non blocking connect waiting result
2004/09/13 12:58:37 BGP: 172.16.22.2 [FSM] TCP_connection_open (Connect->OpenSent)
2004/09/13 12:58:37 BGP: 172.16.22.2 [FSM] Receive_OPEN_message (OpenSent->OpenConfirm)
2004/09/13 12:58:37 BGP: 172.16.22.2 [FSM] Receive_KEEPA_LIVE_message (OpenConfirm->Established)
2004/09/13 12:58:37 BGP: 172.16.22.2 [FSM] Receive_KEEPA_LIVE_message (Established->Established)
2004/09/13 12:58:37 BGP: 172.16.22.2 [FSM] Receive_UPDATE_message (Established->Established)
2004/09/13 12:58:37 BGP: 172.16.22.2 [FSM] Receive_UPDATE_message (Established->Established)
```

You will see in this portion of the log file the sequence of events that happen after the interface is disconnected. In this implementation of BGP the hold timer is three times the KEEPA\_LIVE timer. The hold timer is a mechanism that specifies how long routing information can be held on to without having receive an UPDATE or KEEPA\_LIVE message. If the hold timer expires, the peer router that the timer has expired for is assumed to be down, and the routing information received from it is now invalid.

You will see the behaviour of the hold timer being three times longer than the KEEPA\_LIVE timer, because you'll see this timer expire twice without receiving a new KEEPA\_LIVE message. The third time, after 180 seconds or 3 minutes, you'll see the hold timer expire.

You will notice the status of the BGP session has now changed since the hold timer expired. You will see the message (Established -> Idle). The BGP session is no

longer active and the relationship is considered idle. There is now no active communication between these two peers since one of the routers is considered down.

Afterwards you will see the activity where the router tries to re-connect, it appears that there is another timer called the start timer, and when this timer expires after 10 seconds, the router attempts to re-establish the BGP session, and move away from idle to the connect state.

We see a non-blocking connect attempted, where the source router tries to write to the socket of the destination router. If nothing is returned within a certain amount of time - in this implementation it appears to be 3 seconds - the connect timer will expire again.

It appears that the state of the BGP session has to go to the "Active" state before it can be returned back to the "Connect" state, and once again try to re-establish the session. We see the connection process try again another time, going through the same steps of trying to connect to the socket in a non-blocking manner. The 3 second connect timer expires, the state changes to Connect->Active, then from there to Active->Connect.

Finally at the next attempt the non-blocking connection attempt receives a response while trying to connect to the socket, and the TCP connection is considered open. We see this in the `TCP_connection_open (Connect->OpenSent)` message, and the notification that the connect state has now been moved up to the OpenSent state, showing that the connection attempt succeeded and the TCP connection between the routers has now been established.

Since the source router has already connected and sent an OPEN message, it now needs to receive an open message from the destination router, and then the connection will be confirmed. We see this in the `Receive_OPEN_message (OpenSent->OpenConfirm)` message, and we see the BGP session state moved from OpenSent to OpenConfirm.

Now all that is left is the exchange of KEEPALIVE messages, and then the BGP session can be moved back to the Established state, and shows full communication between the BGP routers is now possible. All of these steps are completed within a second, so you can see that the session establishment is a pretty quick affair. The routers are now established, so the final stage in establishing adjacency is for the routers to exchange their routing information.

## **Routing Information, Conclusions and Observations**

After observing the resultant routing information of our BGP daemon, it was clear that the routers' knowledge was incomplete. Some internal network addresses were IP reachable belonging to peer AS. Also, some prefixes were imported into the routing knowledge of BGP from these peers. However, certain path attributes were absent in the entries for these network prefixes (`AS_Path` & `Next_hop`), and ICMP failures occurred for certain interfaces and not others. When discussing this problem with classmates, we surmised that it was a failure of our interior routing protocol OSPF. We estimated that OSPF was not able convey key information relevant to BGP, at least in a manner that our daemon could understand.

In actual fact OSPF is able to convey this information as part of its design, however this extension is not widely implemented. Another solution was available. There was need to implement iBGP and compliment our IGP with the ability to propogate BGP information across AS. To do this, some understanding of how iBGP could do this was necessary. Inside our AS, we would need to peer in a full mesh configuration with each of our BGP speakers. With this in place, EBGP information would be able to traverse our AS leaving the pertinent path attribute information in tact. Also, next hop information would have to be manually configured in our routing daemons, so that this path attribute could be conveyed by BGP routers passing along information. Remembering our original configuration for BGP, and the addition of iBGP information, the resultant would look like as follows. (Router 3)

```
router bgp 65001
  bgp router-id 1.1.1.1
  network 192.168.1.0/24
  redistribute ospf
  neighbour 172.16.12.7 remote-as 65007
  neighbour 192.168.1.1 remote-as 65001
  neighbour 192.168.1.1 next hop self
  neighbour 192.168.1.2 remote-as 65001
  neighbour 192.168.1.2 next hop self
```

The bolded lines are the new configuration additions, and to generate iBGP traffic inside our AS, the other routers would have to be configured in a similar fashion. Perhaps the most perplexing element of these new entries is the fact that the new peers share the same AS number, as a rule, this is how BGP is aware of the fact that these speakers wish to implement iBGP.

In addition to these changes made in our BGP daemon, some changes should be made to the OSPF router as well.

#### ospfd.conf

```
router ospf
  ospf router-id 192.168.1.1
  redistribute bgp ← ----- Remove
  network 192.168.1.0/24 area 0.0.0.0
```

The bolded line should be removed from this configuration. This should be done to prevent overwhelming the OSPF routers from incorporating all learned routes from BGP into OSPF. This would create a flood of thousands of OSPF messages within the AS. It is necessary to inform OSPF machines of an exit point or points for the AS. This can be achieved in a number of ways, but is most easily accomplished with the Default Gateway Originate command in the OSPF configuration.

Now the routing knowledge of BGP was certainly more complete, and IP reachability greatly improved.