Programming Assignment 3
Due Feb 24 2014 23:00PM
Electronic submissions only.

THE WORK MUST BE YOUR OWN!
If you are referencing any outside sources you must cite them clearly.

In this assignment you will write a program that does simplified BGP route advertising. Your program will take four command-line arguments:
<port number to listen on> <AS number> <BGP ID> <routing table file>

It will first read the routes from the routing table file. Then, it will listen on the specified TCP port and accept connections from BGP peers. When a peer connects, it will send BGP UPDATE messages to the new peer, advertising the routes currently in the routing table. Additionally, it will periodically check the modification time of the routing table file and if it detects that it has changed, it will re-read it and send BGP UPDATEs reflecting the changes to all connected peers. This functionality is described in more detail below.

1. Routing table file
The file format will be very similar to the one in assignment 2, except that the nexthop value will be an actual IP address. That is, the format will be:
<subnet> <netmask> <nexthop>
where all three values are in IPv4 dotted decimal and are seperated by spaces. Example:
112.23.56.0 255.255.255.0 1.2.3.4
Note that the nexthop address is from the point of view of routers outside of your AS.

Another difference from the previous assignment is that the routing table may change after your program has started and you need to be able to detect changes. The simplest way to do that is to periodically check the file modification time and compare it against a previously stored modification time. If the time changes, that means that the file has been changed and you should re-read it and send an update to the currently connected peers.

To check a file's modification time you can use “os.path.getmtime(<file path>)” (after importing the “os” module). You can check the time every 10 seconds (or less for testing) and you should only re-read the file if the time has changed.

2. Communication with BGP peers
When a new client connects to your server, you should first send a BGP OPEN message with your AS number and BGP ID and no parameters (see next section for more details on the message format). You should also try to read a BGP OPEN message from the newly connected peer. If the message you read is invalid (e.g. does not start with marker = 16 FF's or the type field is not set to 1 or the length is incorrect for a BGP OPEN with no parameters) you should print a warning and disconnect the client. If the remote peer's AS or ID is the same as yours, you should also print a warning and disconnect the client.

After the connection has been established an you have exchanged OPEN messages, you should send a BGP UPDATE messages to the peer, advertising all of the routes currently in your routing table. You

will send a separate UPDATE message for each router in the table. The update should include:

- The AS path, which for the purposes of the assignment will be an AS_SEQUENCE with only 1 element: your own AS.
- The next hop address specified in the routing table
- All of the prefixes of the destination networks reachable through that router.

See next section for more details on the message format.

Whenever, the routing table changes (see previous section), you should send an UPDATEs to the peer reflecting the changes. Any destinations that are no longer in the file should be specified in the "withdrawn routes" section. For any routes, that have been changed or added since last time, you should follow the same procedure as specified above for sending update to new peers, except that in this case you will only include the routes that have been changed or added since last time.

3. **BGP Message Format**

   The BGP OPEN message, as discussed in class, looks like this:

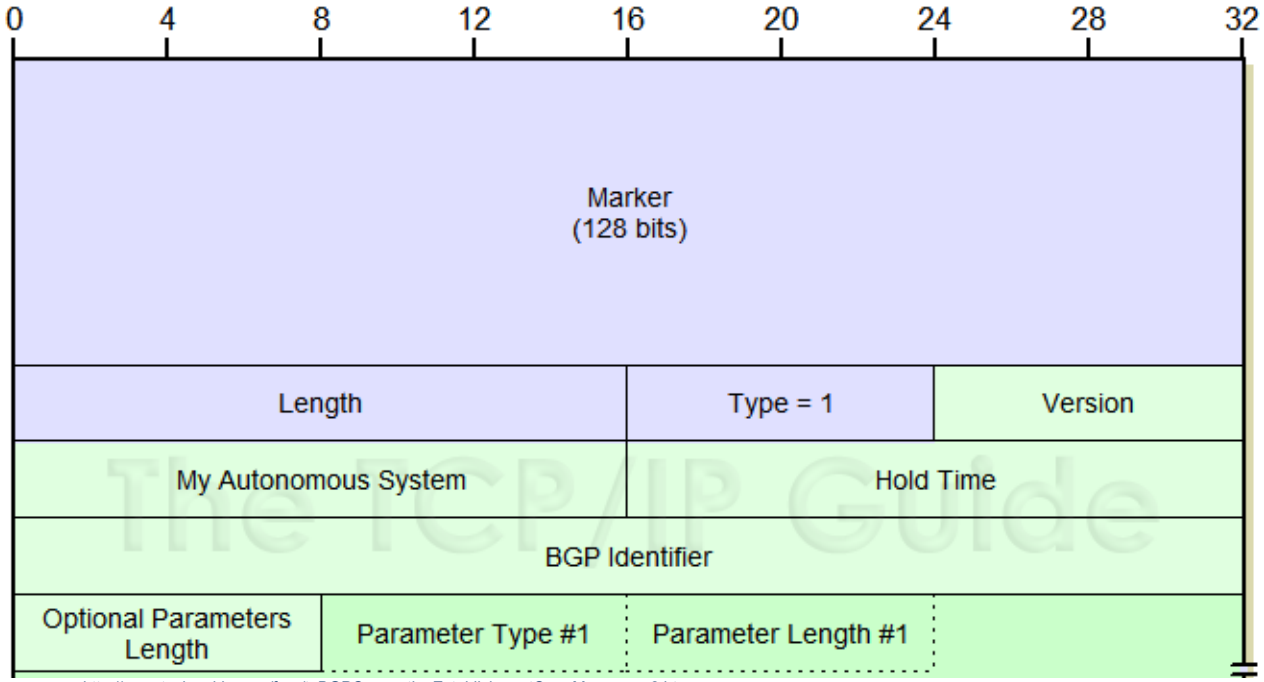

Image source: http://www.tcpipguide.com/free/t_BGPConnectionEstablishmentOpenMessages-2.htm

The marker should be all 1's (i.e. 16 0xFF bytes). The length is the entire length of the message. The type is 1 for type OPEN. For Version, you should use 4. For "My Autonomous System" and "BGP Identifier " you should put the AS number and BGP ID that were specified as command-line arguments. Note that the identifier is in the form of an IP address, so when given to your program, it will be in dotted decimal (i.e. 1.2.3.4) and when you put it in the packet field, it should be a 4-byte network-order integer. You will not be using the hold time, so you can just put 0 there. Also, there will be no optional parameters, so you will want to put 0 for the "Optional parameters length" as well. Note that, as always, all multi-byte fields are in network-order.

The BGP UPDATE message looks like this:

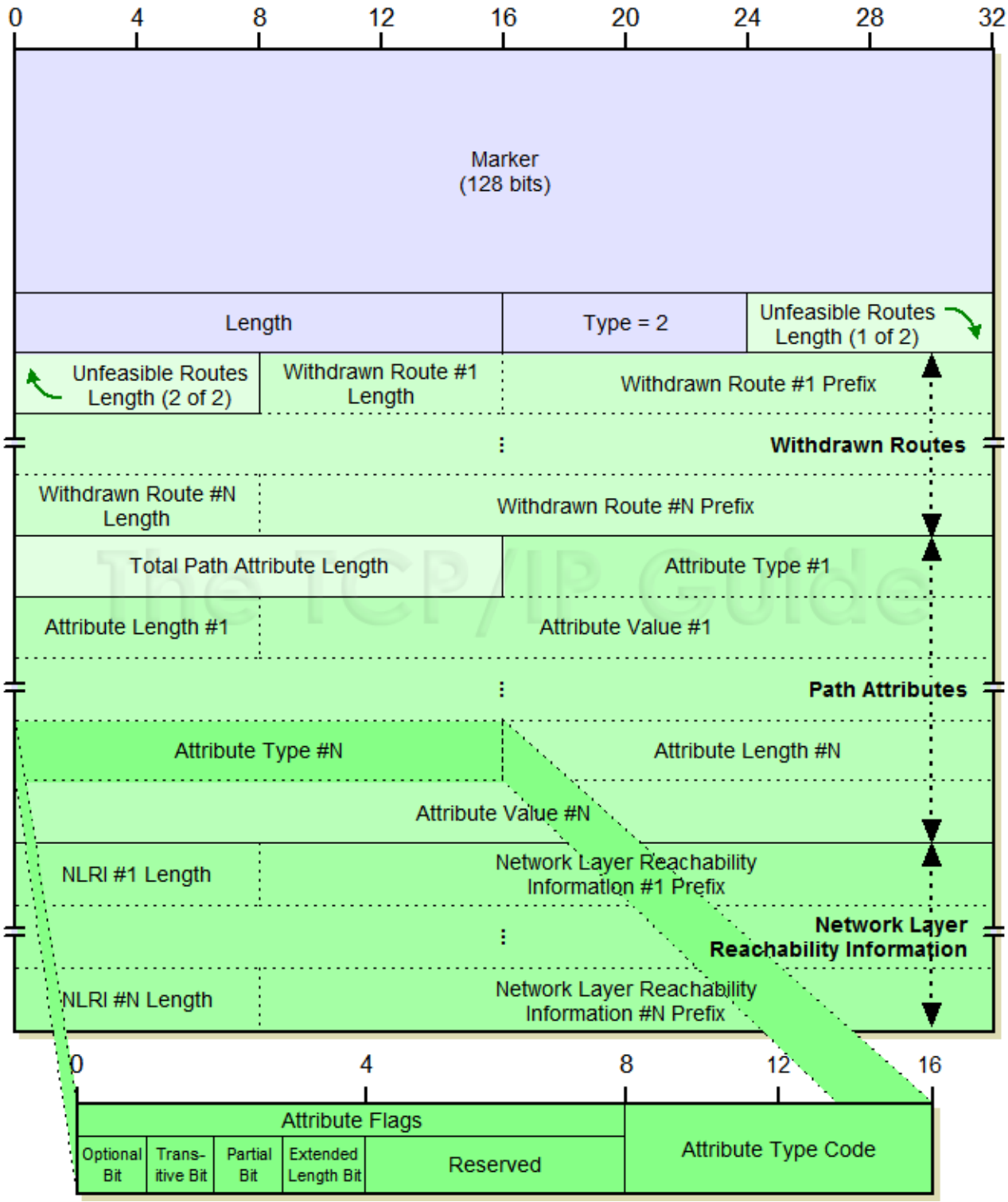

Image source: http://www.tcpipguide.com/free/t_BGPConnectionEstablishmentOpenMessages-2.htm

The “Withdrawn routes” contain prefixes of destinations that have been previously advertised but are no longer reachable through your AS.

The only attributes that you will be using are the AS_PATH (type code 2) and NEXT_HOP (type code 3). For the flags you can always use 0x40. The Attribute Length is the is the total length of the attribute

value. For NEXT_HOP the length is 4 and the value is the IP address of the next hop, encoded in the usual 4-byte network-order format. For the AS_PATH value, you have three fields: path segment type (1 byte), path segment length (1 byte) and list of AS numbers for Autonomous Systems on the path (2 bytes each). In your case the path segment type will always be AS_SEQUENCE (type 2), the path segment length will always be 1 (since you will only be advertising destinations within your AS) and there will be only one AS number (that is, your AS number).

Finally, the Network Layer Reachability Information section includes prefixes of new destinations that you are advertising.

Remember, that, as discussed in class, BGP prefixes (both in the "Withdrawn routes" and the "Network Layer Reachability Information" sections above) are encoded in a compressed format that includes: 1 byte for the number of bits in the netmask set to 1 and between 1 and 4 bytes for the part of the IP address corresponding to the prefix. The actual number of bytes is the minimum that can encode the prefix size. For example: 10.10.3.0/24 would be encoded in 4 bytes: 24, 10, 10, 3. Whereas 172.16.0.0/30 would be encoded as 5 bytes: 30, 172, 16, 0, 0.

It is **strongly** recommended that you first write functions that deal with encoding the data into the proper BGP messages, test them well by themselves and only after that write your socket code for actually communicating with peers at which point you will call the functions that construct the messages. This will save you a **significant** amount of time because debugging will be much quicker that way. To help you with this part, some sample messages are included in next section.

4. Sample BGP Messages

BGP OPEN with AS = 3100 and ID = 10.20.30.40
ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff 00 1e 01 04 0c 1c 00 00 0a 14 1e 28 00 00

BGP UPDATE advertising destination 10.30.1.0/24 with nexthop 10.1.200.45 and AS 100:
ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff 00 29 02 00 00 00 0e 40 02 04 02 01 00 64 40 03 04 0a 01 c8 2d 18 0a 1e 01

BGP UPDATE advertising destinations 172.16.0.0/30 and 172.16.0.4/30 with nexthop 10.20.30.40 and AS 45100:
ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff 00 2f 02 00 00 00 0e 40 02 04 02 01 b0 2c 40 03 04 0a 14 1e 28 1e ac 10 00 00 1e ac 10 00 04

BGP UPDATE advertising destinations 10.30.1.0/24, 10.30.2.0/24 and 10.30.3.0/24 with nexthop 10.1.200.45 and AS 5100:
ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff 00 31 02 00 00 00 0e 40 02 04 02 01 13 ec 40 03 04 0a 01 c8 2d 18 0a 1e 01 18 0a 1e 02 18 0a 1e 03

BGP UPDATE withdrawing destinations 10.30.1.0/24 and 10.30.2.0/24
ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff 00 1f 02 00 08 18 0a 1e 01 18 0a 1e 02 00 00