# MRT User/Configuration Guide

## *Version 2.0.0 Alpha*

*(Draft 11/5/99*

# Table of Contents

# Copyright (c) 1997, 1998, 1999

# Introduction

This chapter introduces the *MRT User/Configuration Guide* and explains how to obtain further information about MRT.

## Document Conventions

The following document conventions are used in the *User Guide:*

- Commands and keywords are in **boldface**.
- User-supplied variables are enclosed in <angle brackets>.
- Optional elements are shown in [square brackets].
- Alternative but required keywords are grouped in {braces} and separated by a vertical bar.

## Related MRT Manuals

The following additional documentation is available for MRT users (see (http://www.merit.edu/net-research/mrt/html/mrt_doc/):

- *Installation Guide*
- *Programmer's Manual*
- Tutorial (in preparation)

The MRT web site will also have the most up-to-date documentation and code.

## Getting Help

For more information about MRT, send mail to mrt-support@merit.edu.

The MRT development team is available to answer questions and provide configuration advice. We are also very interested in bug reports, feature requests, and general feedback.

A mailing list, mrt-discuss-request@merit.edu, is also available for MRT users to share advice and experiences with the toolkit.

## Credits

MRT was originally developed by Merit Network, Inc., under National Science Foundation grant NCR-9318902, "Experimentation with Routing Technology to be Used for Inter-Domain Routing in the Internet." The current research is supported by the National Science Foundation (NCR-

9710176) and a gift from Intel Corporation.

The design and ideas behind many of the MRT libraries draws heavily on the architecture pioneered in the GateD routing daemon.

The University of Michigan/Merit Network MRT development team includes: Craig Labovitz, Masaki Hirabaru, Farnam Jahanian, Susan Hares and Susan Rebecca Harris. Additional code and architecture ideas were supplied by Marc Unangst and John Scudder.

Francis Dupont developed the initial BGP4+ code.

The public domain Struct C-library of linked list, hash table and memory allocation routines was developed by Jonathan Dekock <dekock@cadence.com>.

David Ward <dward@netstar.com> provided bug fixes and helpful suggestions.

Pedro Roque developed the first port to Linux IPv6, and wrote many of the Linux kernel interface routines.

Mark Turner <mturner@cisco.com> added several new features to BGPSim.

We would also like to thank our other colleagues in Japan, Portugal, the Netherlands, the UK, and the US for their many contributions to the MRT development effort.

# 1. Overview

The MRT toolkit has been used to build a wide variety of tools, ranging from production Internet and 6bone routing daemons to BGP fault-injection and traffic generation test packages. MRT software is in active use at universities and commercial organizations throughout the country and internationally.

MRT uses novel approaches to routing architecture design, and incorporates features such as parallel lightweight processes, multiple processor support, and shared memory. The object-oriented, modular design of the software encourages the rapid addition and prototyping of experimental routing protocol and inter-domain policy algorithms.

Although MRT has been designed with multi-threaded, multi-processor architectures in mind, the software will run in emulation mode on non-thread capable operating systems.

Today, MRT applications and libraries enjoy a diverse user community, researchers and commercial developers across the Internet. See the **Tutorial** for examples on how users are adopting MRT technology.

You can use MRT applications and libraries to:

- Serve as the backbone routing software for your IPv6 or IPv4 network connection.
- Simultaneously handle tasks such as routing policy communication, routing policy calculation, and maintenance of a RIB, and distribute these tasks over multiple processors or multiple machines
- Generate and analyze route flap statistics
- Generate real-time graphical maps of Internet routing
- Capture a BGP peering session and monitor it in real time
- Record and replay sequences of events, such as routing failures

The toolkit includes two main categories of tools: Routing and Network Performance measurement tools. A high level description of the toolkit applications is provided below:

**Routing Tools**

- **MRTd—**A routing daemon supporting RIPng, BGP4+, multiple RIBs (route server), and RIP1/2. MRTd reads Cisco Systems-like router configuration files and supports a Cisco Systems router-like telnet interface.
- **BGPsim—**A BGP4+ traffic generator/simulator.
- **SBGP—**A simple BGP4+ speaker and listener**.**

- **Route_BtoA—**Converts binary MRT messages to ASCII**.**
- **Route_AtoB—**Converts ASCII descriptions of MRT messages to binary MRT message format. Binary MRT messages can be piped into other MRT programs, including SBGP and BGPSim.

# 2. Getting Started

See the **MRT Installation Guide** for information on building and installing MRT programs

All MRT programs can be invoked from the command line, or from the Unix boot/startup script. Below is an example of starting the MRTd routing daemon from the command line:

```
> /usr/local/bin/mrtd
```

Once running, most MRT-based tools will begin to listen for user telnet connections on the TCP port specified in /etc/services. MRT-based programs may be configured by editing a configuration file, or by invoking the configuration utility from the interactive user telnet interface. Below is an example of telneting to the user interactive interface (UII) port on a machine running the MRTd routing daemon. The “mrtd” number has been configured in /etc/services (see the Installation Manual for more information).

```
>telnet 127.0.0.1 mrtd
MRT version 1.5.2 ALPHA February 22, 1999
User Access Verification
[71] password> ***
[71] MRTd>
```

If a password is specified in the configuration file, it must be supplied at the password prompt. Initially, MRT programs default to no password access control and restrict user interactive telnet to the loopback address or the interface address of the local machine.

The MRT user interface supports Unix shell-like redirection (> or >> **filename**) for output. To edit a line, emacs-like line editing, including ^a, ^e, ^b, ^f, ^d, ^k, ^u and ^c, is available. To reuse a previous line, the tcsh-line history function is available by typing ^p and ^n.

Most MRT-based tools share a common subset of user management and configuration commands. The command language used by MRT shares many similarities with the language used on Cisco Systems routers.

Commands common to most MRT tools include:

- **show config—**view the configuration file
- **show version—**show the current version
- **show threads—**show the status of application threads
- **config*****—enter configuration mode
- **enable**—enter enable mode
- **write*****—save volatile memory configuration to disk

- **reboot*****—restart the application
- **help—**shows all commands available
- **exit**—leave the UII interface

***** Note that if you set 'enable password' in your configuration, the enable command is required in order to execute some potentially dangerous commands, such as config, write, and reboot (marked with an asterisk * above.)

## Configuration Commands

When MRT programs are started for the first time and no configuration file exists on disk, the programs will create a default configuration in volatile memory. This configuration may be modified in memory by issuing the "**config"** command from the UII telnet interface prompt. Modifications to volatile memory may be saved to disk using the "**write**" command. Modifications not saved to disk will be lost if the application terminates or is rebooted.

Upon startup, MRT programs will search for the default configuration file for the application (usually /etc/<application_name.conf>). The user may also override the default configuration file by providing a "**-f <filename>** " flag on the command line of the application.

Beginning with this version (v. 1.5.1 alpha), all configuration commands may be issued even through the interactive, telnet interface. Configuration changes can also be made directly to the configuration file on disk. If changes are made the program must be restarted, or **rebooted**, to reread the changed configuration file.

Most configuration commands have no-style commands, such as **no ip route .....** to remove them from the configuration. Command prefixes can be used for both IPv4 and IPv6 configurations, but IPv6 features may not be available on IPv4-only platforms.

The following configuration commands are common to most MRT-based programs:

**line vty—**configures the user interface (by telnet)

**password <string>**

Sets a password **<string>** for telnet interface. Note that if a password is not set, access verification will not be performed and interactive user telnet connections will only be available from the local host.

**access-class <access-list number>**

If **<access-list number>** is specified, telnet connections will be restricted to IP addresses allowed by the access list. See the **access-list** description below for more information.

**port <number>**

Changes the port number with **<number>** for the telnet interface. The default is the port value specified in /etc/services for the application name. If a /etc/services entry does not exist, the port number assigned to the service defaults to "mrt" or 5674.

**exec-timeout** <**minutes**>[<**seconds**>]

Sets the timeout on idle UII connections to the number of <minutes> and **<seconds>.** At the end of the timeout interval, the idle UII connection is closed.

**addr** <**address**>

Specifies the address at which the UII connections are accepted. The default is all addresses available on the host.

**login**

Enables connections from other hosts.

## **enable password—**sets enable password

**enable password <string>**

Sets enable password <string> for the enable command.

## **debug**—controls debug options

**debug** <**flag**> **[**<**file**> **[<size>]]**

Logs debug messages matching priority <**flag**> to the named <**file**>. If the file size limit is specified with <**size**>, the file will be truncated after reaching the <**size**> byte limit. If <**file**> is omitted, stderr will be selected.

**<flag> .... info, norm, trace, parse, packet, state, timer, all**
**<file> .... filename or "stdout".**
**Some applications also support "syslog"**

**<size> .... maximum size (bytes) of logfile.**
**File is truncated and restarted after reaching <size>.**

**NOTE:** Debugging may significantly impact the performance of mrtd and other daemons. We recommend using binary packet dump update option for statistics collection and event logging.

## **access-list—**defines a filter

**access-list** <**number**> **{permit|deny}** <**prefix**> **[refine|exact]**

Defines an access list <**number**>, which permits or denies access if the condition is matched. **all** can be specified as <**prefix**>. **exact** will be assumed if neither **refine** or **exact** is specified. **exact** matches only the prefix, while **refine** matches more specific prefixes, excluding the prefix itself.

Matches are performed in the order in which they appear. At the end of a list with the same number, deny all is assumed.

**!—**comment and separator

> Comments can appear at the beginning of a line, or any other place in the line. However, only comments appearing at the beginning of a line are stored in memory.

**redirect**—allows shell-like redirection of output (> or >>).

> **redirect** <directory>
>
> > Allows redirection to files in this directory. Unrestricted redirection was deemed a security problem.

After editing the configuration file, the user may return to the top-level of the interactive telnet interface by typing a **^Z** or entering **exit**. Below is an example of an interactive telnet session using the configuration mode of an MRT application:

```
Config> ?
  access-list
  as-path
  debug
  dump
  dump-binary
  exit                          Quit from the current level
  gateway
  interface
  no
  password
  quit                          Quit from the current level
  redirect
  route
  route-map
  router
  show
```

## Combining MRT Programs

Many MRT programs (sbgp, route_btoa, bgpsim) can be combined together using Unix shell-like pipe features. For example:

```
sbgp -o stdout | route_btoa -i stdin
```

Programs can also use System5 message passing with -r and -w options. Note that this feature is limited/experimental in this release.

## Tracing/Logging

All MRT programs support logging. The **–v** command line option will turn on verbose logging to stdout. More advanced logging options may be set in the configuration file using the **debug** command.

**NOTE:** Debugging may significantly impact the performance of mrtd and other daemons. We recommend using binary packet dump update option for statistics collection and event logging.

MRT supports five different debugging flags: NORM, PACKET, PARSE, TRACE, STATE, PARSE, POLICY, and ALL.

| | |
|---|---|
| NORM | Logging from normal operations. Includes error messages and significant events |
| PACKET | Logging of receipt and transmission of packets (e.g., BGP updates) |
| PARSE | Logging of config file parsing and UII configuration changes |
| POLICY | Application of filter lists or routing policy. |
| STATE | Protocol state machine changes and other events (e.g., BGP state machine changes) |
| TRACE | Detailed logging of operations |
| ALL | All logging turned on |

MRT debugging information may be forwarded to the UII vtty by issuing the command **terminal monitor**.

# 3. MRTd

MRTd is a multi-threaded routing daemon with support for BGP4, RIP1/2, RIPng, and BGP4+ (for IPv6) and multiple RIBs (i.e. route server). MRTd reads Cisco Systems-like router configuration files and includes a Cisco Systems router-like interactive telnet interface.

At the moment, BGP, RIPng, and BGP4+ are completely supported. RIP1/2 and its interaction with other protocols are not supported. Development of OSPF and PIM-DM is currently underway.

## Synopsis

**mrtd [-v] [-n] [-f configuration file] [-l rib file] [-r] [-m]**

## Options

**-f configuration_file**
Read the specified configuration file. By default, MRT tries to read /etc/mrtd.conf.

**-v**
Turn on verbose logging to standard output. This is useful to turn on logging before the debug commands are read in the configuration file.

**-n**
Specify that MRT will not modify the kernel routing table. (This option is used to test MRT configurations with actual routing data.)

**-r**
Don't install routes in the RIB.

**-l routing database**
Load routes from a routing table dump and use these prefixes in the simulation. The routing database file must be in MRT message format. The programs included in the route_atob directory will convert GateD, RSd and Cisco Systems routing table dumps to MRT RIB format.

**-m**
Use a new dump format.

## Description

MRT first reads its configuration file (by default /etc/mrtd.conf) to configure routing protocols, route peerings, and routing policy. The configuration file closely resembles those used by Cisco Systems routers.

After reading the configuration file, MRT scans the kernel for existing routes, scans the kernel interface list, and then initiates routing protocol communications. MRT also begins listening on the mrt service port, "mrtd," (specified in /etc/services) for user telnet connections.

## Sample Configuration Files

A sample IPv4 MRT configuration file is shown below:

```
Line vty
   password my_password
   login
!
debug norm stdout
!
access-list 1 deny 0.0.0.0/0
access-list 1 permit all
access-list 2 permit 192.168.0.0/16
!
router bgp 185
neighbor 192.168.10.2 remote-as 65
neighbor 192.168.10.2 distribute-list 1 in
neighbor 192.168.10.2 distribute-list 2 out
neighbor 198.108.60.244 remote-as 185
redistribute static
redistribute rip
!
router rip
network 192.168.10.0/24
network 198.108.60.0/24
redistribute static
redistribute bgp
!
ip route 192.168.100.0/24 192.168.10.100
ip route 192.168.150.0/23 192.168.10.100
ip route 192.168.190.1/24 192.168.10.100
ip route 10.0.0.0/8 192.168.10.100
```

Following is a sample IPv6 configuration file:

```
Line vty
   password my_passwordhttp://pythia.uoregon.edu/~llynch/nanog16.html
   port 5674
   login
!

dump bgp view 1 /susr/masaki/tmp/ipv6/bgp.routes.%y%m%d.%H:%M 60m

dump bgp updates /susr/masaki/tmp/ipv6/bgp.updates.%y%m%d.%H:%M 15m

debug all /tmp/MRTd.log 1000000

redirect /tmp

!

access-list 1 deny 3ffe:1c00::/24 refine !merit internal
access-list 1 permit all
!
access-list 99 deny all
!
as-path access-list 1 permit ^1673 ! just an example, it's always true
!
router bgp 237 ! define own AS number
  aggregate-address 3ffe:1c00::/24 summary-only as-set
  neighbor 3ffe:0dfe:fffe::9 remote-as 1673 ! eBGP
  neighbor 3ffe:0dfe:fffe::9 description ANS
  neighbor 3ffe:0dfe:fffe::9 distribute-list 1 out ! drop specific
  neighbor 3ffe:0dfe:fffe::9 filter-list 1 in ! as path filter
  neighbor 3ffe:1c00::3 remote-as 237 ! iBGP
  neighbor 3ffe:1c00::3 description CC
  neighbor 3ffe:1c00::3 bgp4+ 1 ! use RFC version of BGP4 MP
  neighbor 3ffe:1c00:0:60::112 remote-as 112 ! for test
  neighbor 3ffe:1c00:0:60::112 remote-as 112 description MRT
  neighbor 3ffe:1c00:0:60::112 distribute-list 99 in ! drop everything
```

```
  neighbor 3ffe:1c00:0:60::112 bgp4+ 1 ! ! use RFC version of BGP4 MP
  redistribute static ! inject static routes
  redistribute direct ! inject connected routes
!
router ripng
  network 3ffe:1c00:0:60::/64
  network 3ffe:1c00:0:12::/64
  network cti1
  network cti2
  redistribute static
  redistribute direct
  redistribute bgp
  distribute-list 99 in cti2
  distribute-list 99 out cti2
!
ip route 0.0.0.0/0 198.108.60.1 ! default route
ip route 3ffe:1c00::/24 ::1 ! merit pTLA
```

## Configuration Guide

This section introduces the command sets for:

- Configuring MRTd and BGPsim
- Using MRT's interactive interface to monitor the status of MRTd and BGPsim.

For information about the **uii, debug,** and **access-list** configuration commands, see Chapter 2, "Getting Started."

### Configuration Commands

As mentioned in Chapter 2, all of the below options may be configured directly through the UII telnet interface. Administrators may also choose to edit the configuration file directly on disk. In this case, MRTd must be **rebooted** before the changes will take affect.

#### Configuring Routes and Policy

MRTd supports most of the common Cisco Systems routing policy commands, including access lists, as-path access lists and route maps.

**route—**defines a static route

**route <prefix> <next hop> [<interface>]**

Establishes a static route to a destination **<prefix>** via **<next hop>**. **<next hop>** may be an IPv4 or IPv6 address and must be consistent with **<prefix>**. To use a specific interface, specify **<interface>**.

**as-path access-list**—defines an as-path access-list

**as-path access-list <number> {permit|deny} <as-regular-expression>**

Defines an as-path access-list <number>, which permits or denies access if <as-regular-expression> is matched.

Matches are performed in the order in which they appear. At the end of a list with the same number, deny .* is assumed.

The as regular expressions are as follows:

<number> an as number (1 through 65535)

| | |
|---|---|
| . | Matches any single as number |
| * | Matches 0 or more sequences of the pattern |
| + | Matches 1 or more sequences of the pattern |
| ? | Matches 0 or 1 occurrences of the pattern |
| ^ | Matches the beginning of the as path |
| $ | Matches the end of the as path |
| \| | Matches one of the alternatives |
| () | Encloses a pattern |

For example:

| | |
|---|---|
| .* | any AS path, including null |
| 237$ | originated from AS 237 |
| 237 | via AS 237 |
| ^(237\|10) | from AS 237 or AS 10 |
| ^$ | originated from this AS |

**network <prefix>**

Configure routes originating in BGP.

**route-map—**define a route-map

**route-map <number>**

Defines the conditions to modify attributes of any updates.

**route-map** sub commands

**set as-path [prepend] <as-path-string>**

Sets or prepends **<as-path-string>** to the as-path of the route. Note that there is no matching function implemented.

**<as-path-string> ... <number>... a sequence of AS numbers**

**[ <number>... ] a set of AS numbers**

**set community [additive] (<number>|no-export|no-advertise)**

Sets community attribute to the route, or appends if additive is specified.
**<number>** values are 1 to 4294967200.

**set origin (igp|egp <as>|incomplete)**

Sets the origin code.

**set next-hop <address>**

Sets the nexthop attribute. If the address is an IPv6 global address, it is set as BGP4+ next hop. In addition, If the address is an IPv6 link-local address, it is set as BGP4+ next hop link local address.

**set metric <number>**

Sets the metric value (MED).

**set local-preference <number>**

Sets the local preference value.

**set dpa as <number> <number>**

Sets the DPA values.

**set atomic-aggregate**

Sets automatic aggregate attribute.

**set aggregator as <number>** <address>

Sets aggregator information. <address> should be IPv4.

### Configuring Routing Protocols

MRTd supports BGP4, BGP4+, RIP2, and RIPng. This version includes partial support for OSPF, and a PIM-DM implementation is underway.

**router—**configures routing protocol

**router bgp** <as number>

Enables assignment of the BGP (or BGP4+ if IPv6 is available) routing protocol **<as number>** to the routing process.

**router ripng**

Enables RIPng routing protocol.

**router rip**

Enables RIP routing protocol.

The following commands are available for the **router bgp** command.

**neighbor <peer address> remote-as** <peer's as number>

Adds an entry of BGP neighbor with **<peer address>**.
<peer's as number> should be an AS number to which the peer belongs.
Must precede other neighbor commands for **<peer address>.**

**neighbor <peer address> update-source <source address >**

Specifies the addresses for outgoing BGP connections and at which incoming BGP connections are accepted.

**neighbor <peer address> next-hop-self**

Forces the next hop in the AS path to be the host itself.

**neighbor** <peer address> **(transparent-as|transparent-nexthop**)

Set transparent option for neighbor, as in use as a route server.

**neighbor** <peer address> **passive**

Does not initiate BGP connections—only accepts them.

**neighbor <peer address> maximum-prefix <number >**

Sets the maximum number of prefixes incuded in a BGP update.

**neighbor <peer address> distribute-list <number> {in|out}**

Applies access-list **<number>** to incoming (in) or outgoing (out) route updates for a peer with **<peer address>**.

**neighbor** <peer address> **filter-list** <number> **{in|out}**

Applies as-path access-list <number> to incoming (in) or outgoing (out) route updates for a peer with <peer address>.

**neighbor** <peer address> **weight** <num>

Set a weight associated with a peer.

**neighbor** <peer address> **trace**

Enable tracing of a BGP peer.

**neighbor <peer address> route-map <number> {in|out}**

Applies a route-map **<number>** to incoming (in) or outgoing (out) route updates for a peer with **<peer address>**.

**neighbor <peer address> route-reflector-client**

Sends routes to an internal peer even if learned from another internal peer (route reflection.)

**neighbor <peer address> description** <string>

Attaches <string> to the neighbor as a description.

**neighbor <peer address> (holdtime|keepalive|connectretry|starttime) <num>**

Set the timer for a neighbor.

**neighbor** <name> **neighbor-list** <num>

Allows anonymous neighbor peers.

**neighbor <peer address> bgp4+ (0|1|old|new|rfc|auto)**

Specifies BGP4+ packet format. The default is 0.

**redistribute <proto>**

Redistributes routes from **<proto>** such as rip to BGP.

**aggregate-address <prefix> [summary-only] [as-set]**

Creates an aggregate entry to **<prefix>**. summary-only suppresses all more specific routes from updates. as-set merges as paths to generate as-set path attribute.

**bgp router <id>**

Defines the router ID used in BGP. The router ID should be an IPv4 address assigned to the host. The default is one of the addresses available on the host; which is picked up by MRT automatically.

**bgp cluster-id <id>**

Defines the cluster ID used in the BGP reflector. The default is the same as the router ID.

## **router rip/ripng**—RIP/RIPng routing

The following commands are available for the **router RIP/RIPng** command.

**network {<prefix>|<interface>}**

Specifies interface(s) by **<prefix>** or by name. to turn on RIP/RIPng. All interfaces included under **<prefix>** will be enabled.

**distribute-list <number> {in|out} <interface>**

Applies access-list **<number>** to incoming (in) or outgoing (out) route updates on **<interface>**.

**redistribute <proto>**

Redistributes routes from **<proto>** to RIP/RIPng.

### Statistics Collection

MRTd can log both routing table dumps and binary traces of all BGP events in a format parseable by other MRT (and soon Zebra) tools. So, for example, BGP updates can be recorded via MRTd and later replayed to test peers through bgpsim or sbgp. *The MRT Programmer's Manual* includes a description of the MRT packet formats.

## **dump bgp--**dump BGP updates, state changes, and routes

**dump-binary [{ip|ipv6}] bgp routes <filename> [<duration>]**

Dump BGP routing table in binary MRT format. **<filename>** can be in strftime() format. If **<duration>** is specified, the file will be reopened every **<duration>**, re-evaluating the filename. If **ip** or **ipv6** is specified, only the routes of the address will be dumped.

**dump [{ip|ipv6}] bgp {routes|updates|all} <filename> [<duration>]**

Dumps BGP/BGP4+ routes, updates, or all into the file **<filename>**.
**<filename>** can be in strftime() format. If **<duration>** is specified, the file will be reopened every **<duration>**, re-evaluating the filename. If **ip** or **ipv6** is specified, only the routes of the address will be dumped.

**dump bgp view <view number> <filename> [<duration>]**

Dump routing table for specified view.

## Interactive Interface Commands

MRTd and BGPsim provide an interactive user interface for management (e.g., viewing the routing table) and configuration.

The following commands are specific to MRTd and BGPSim. Additional commands are described in Chapter 2, “Getting Started.”

**clear bgp * <name> – Close/reset BGP peering session with this peer ***

**config * – Enter configuration mode**

**quit – Exit mode, or exit UII interface**

**show – show system information**

**show [{ip|ipv6}] bgp**

**show [{ip|ipv6}] bgp summary**

Show BGP peers summary

**show [{ip|ipv6}] bgp neighbors**

Show BGP peers and their status

**show bgp neighbors** (<peer address>|<name>|*) **errors**

Show recent BGP errors/notifications with this peer.

**show bgp neighbors** (<peer address>|<name>|*) **routes**

Show BGP routes sent to this peer

**show [{ip|ipv6}] bgp routes**

Show BGP routing table

**show [{ip|ipv6}] bgp regexp** <as-regular expression>

Show BGP routes matching the as-path regular expression.

**show [{ip|ipv6}] bgp prefix** <prefix>

Show BGP routes matching this prefix.

**show config**

Show the current configuration

**show interfaces**

Show all interfaces available

**show ip**

Show IPv4 routing table

**show ipv6**

Show IPv6 routing table

**show rib**

Show the central routing table

**show rip**

Show RIP status

**show rip routes**

Show RIP routing table

**show ripng**

Show RIPng status

**show ripng routes**

Show RIPng routing table

**show view** <view number>

Show the BGP routing table for this view.

## dump & load – dump and load bgp binary routing table dump to/from disk

**dump [{ip|ipv6}] bgp routes <filename>**

Dumps bgp routes into the file <filename>.

**load [{ip|ipv6}] bgp routes <filename>**

Loads bgp routes from the file <filename>. Note that this will introduce inconsistency into the routing table.

## trace – log protocol information to disk or UII

**trace [{ip|ipv6}] bgp** *

Enable tracing of BGP protocol.

**trace bgp neighbor** (<peer address>|<peer name>) *

Enable tracing on the peer. (The "terminal monitor" command is required to watch this at the UII.)

**trace bgp view (*|inet|inet6|<num>)** *

Enable tracing of view routing table changes.

## quit—quit the mode or disconnect

* Note that if you set 'enable password' in your configuration, the enable command is required in order to execute some potentially dangerous commands, such as clear bgp, config, and trace bgp (marked with an asterisk* above.)

Following are examples of the interactive interface commands:

```
[21] MRTd> show ip

4 prefixes
 P Pref Time     Destination               Next Hop                  If
*S    1 74:42:37 0.0.0.0/0                 198.108.60.1              ep0
*C    0 74:42:37 127.0.0.0/8               0.0.0.0                   lo0
*C    0 74:42:37 192.168.12.0/24           0.0.0.0                   lo0
*C    0 74:42:37 198.108.60.0/24           0.0.0.0                   lo0
```

```
[17] MRTd> show ripng

Routing Protocol is "ripng" (Using IPV6)
Listening on port 521 (socket 10)
Sending updates every 30 seconds +/- 15, next due in 29 seconds
Triggered update and split horizon (no poisoned reverse) implemented
Invalid after 180 seconds, hold down 180, flushed after 120

106 ripng routes and 107 ripng attributes active
106 hash entries
```

# 4. BGPsim

BGPsim simulates complex BGP4 routing environments with possibly high levels of routing instability/change.

BGPsim includes a perl program, BGPsim.pl, which is used to generate ASCII descriptions of BGP traffic for use with route_btoa and sbgp. (The BGPsim Perl code is still quite rough.)

## Synopsis

**BGPsim [-f configuration_file] [–l routing table] [-v] [-s] [-m]**

## Options

**-f configuration_file**
Read the specified configuration file. By default, bgpsim tries to read ./bgpsim.conf.

**-v**
Turn on verbose logging to standard output. This is useful to turn on logging before the debug commands are read in the configuration file.

**-s**
By default, BGPSim does not set mandatory BGP attributes, including origin, nexthop and ASPath. If this flag is used, BGPsim will add these attributes and prepend the local AS to the ASPath.

**-m**
Use a new dump format.

## Description

By default, BGPsim looks for "./bgpsim.conf". The format of the configuration file is described below. BGPsim also has an interactive (Cisco Systems router-like) interface: telnet to port 5674 on the machine running BGPsim.

**NOTE:** BGPsim does not include mandatory attributes by default. You will need to explicitly include a nexthop, origin, and apsath attribute in your BGPsim configuration. Also note that BGPSim does not prepend its own AS by default.

## Sample Configuration File

A sample BGPsim configuration file is shown below.

```
network-list 1
range 10.0.0.0/8 11.0.0.0
stability 10 jitter 4
map 1
!
route-map 1
set nexthop 198.108.60.8
set aspath 185 123 23 23 12
set origin igp
!
```

```
network-list 2
range 192.32.0.0/24 192.32.255.0
stability 9 jitter 3
change 12 jitter 4
route-map 2 3
!
route-map 2
set next-hop 198.108.60.244
set as-path 185 123 23 23 12
set origin igp
set community 56:123
set dpa as 56 121
set local-preference 23
!
route-map 3
set as-path 185 100 10 102
set origin igp
set community 100:345
set dpa as 3 23
set local-preference 83
!
router bgp 185
  neighbor 198.108.60.244 remote-as 65
  neighbor 198.108.60.112 remote-as 165
```

This file describes two simulation processes, as defined by network-list 1 and 2, which changes routes to two BGP peers (AS 65 and AS165).

The first simulation process, network-list 1, changes routes (10.0.0.0/8 and 11.0.0.0/8) as defined in range every 10 seconds. This simulates an announcement of the routes first, and then a withdrawal after 10 seconds. Ten seconds after the withdraw, the next announcement is propagated. Thus the announcements and withdraws are repeated every 10 seconds. These routes have attributes defined in route-map 1: nexthop is 198.108.60.8 and aspath is a sequence of 123 23 23 12.

The second network list describes simulation of the range of routes from 192.32.0.0/24 to 192.32.255.0/24 (i.e. 192.32.1.0/24, 192.32.2.0/24, etc.) All of these routes have an initial aspath of (123 23 23 12), a next-hop of 198.108.60.244, and others as defined in route-map 2. These attributes change every 12 seconds among route-maps 2 and 3.

The peers (AS 65 and AS165) receive routing updates originated by these two simulation processes.

## Configuration Commands

For information about the **uii, debug,** and **access-list** configuration commands, see Chapter 2, “Getting Started.” In addition to the MRTd configuration commands, the following are available in BGPsim to simulate routing changes:

**network-list** <number>

Defines a network-list with **<number>**. This definition behaves like a routing process which generates routing

changes within a range defined by range subcommand by an interval defined by stability subcommand, changing route attributes as specified by change and route-map subcommands.

Options include:

**range <start prefix> <end prefix>**

Defines a range to announce and withdraw, starting with **<start prefix>** up to **<end prefix>** (inclusive). The range is along classful boundaries.

**stability <interval number of seconds>** [**jitter <jitter number>**]

Defines an interval **<number>** in second to change routes. Routes are announced first and then withdrawn after the interval. Thus, with the interval, announce and withdraw **repeat**. Jitter adds/subtracts a random number of seconds between 0 and <jitter number> to the interval.

**change <interval umber of seconds>** [**jitter <jitter number>**]

Defines an interval **<number>** in second to change attributes of routes being announced. route-map subcommand defines a sequence. Jitter adds/subtracts a random <number> of seconds from the timer.

**map <number> ...**

Defines a sequence of route-maps to be used. The next route-mapis adopted after the interval defined in change subcommand. At the end of list, the first route-map is adopted as a next. The first route-map behaves as a default, that is, this is always adopted before adopting other route-maps.

**file <filename> ...**

Loads routes from routing table dump file <filename> rather than using a range of addresses.

BGPSim also adds several commands to bgp router commands:

**neighbor <peer address> stability <seconds>**

Define stability for TCP peering session with this peer.

## Interactive Interface Commands

The BGPsim interactive interface supports the following commands in addition to MRTd interactive interface commands:

**show simulation**

**stop simulation**

**start simulation**

# 5. SBGP

SBGP is a simple BGP4 speaker and listener. SBGP does not apply policy to routes, nor does it maintain a routing information base (RIB) of routes it has previously learned. Rather, SBGP provides a mechanism for monitoring routing information sent from a peer, and for injecting routing information into a peering session.

## Synopsis

**sbgp [-av] [-i binary_data_in_file] [-o binary_data_out_file] [-l log_file] [-f config_file] [-c port] [-d port] [my AS] [peer_IP peer_AS]...**

## Options

**-a**

Accept peering BGP connection from all peers.

**-v**

Turn on verbose logging to standard output.

**-i binary_data_in_file**

Inject routes from this file into every peering session. Use the file name 'stdin' to read input from standard in.

**-o binary_data_out_file**

Save route updates from all peering sessions into this file. Use the file name 'stdout' to write output to standard out.

**-l log_file**

Write logging information to this file. By default, SBGP logs to /tmp/bgp.log.pid, where pid is the process ID number of the SBGP process.

**-f config_file**

Not supported yet

**-p**

Not supported yet

**-c port**

Connect to this port on all BGP peers.

**-d port**

Listen on this port for BGP peering connections.

**[my AS] [peer_IP peer_AS]...**

Use my AS for my Autonymous System number and open peering sessions with each peer_IP address.

## Description

As arguments, SBGP takes the local AS number followed by the IP address and AS number of the BGP4 peer. Multiple peer IP addresses and AS pairs may be specified. For example:

**sbgp AS2011 enss131.t3.ans.net AS690**

attempts to initiate a BGP4 peering session with the old NSFNET backone on enss131. By default, SBGP writes logging information to /tmp/bgp.log.

The following command directs tracing information to stdout (the -v option) and will save MRT messages containing the contents of BPG4 update packets to /tmp/data (the -o option).

**sbgp -vo data AS2011 enss131.t3.ans.net AS690**

Note that the remote peer must be configured to accept a BGP4 peering session from the machine on which SBGP is running.

The following command will inject routes stored in the binary MRT message file data into the peering sessions with enss131:

**sbgp -vi data AS185 enss131.t3.and.net AS690**

# 6. ROUTE_BTOA

ROUTE_BTOA converts binary MRT messages to ASCII. By default, the program writes human-readable ASCII descriptions of MRT message streams or files to standard out. Binary MRT messages may be generated by programs such as SBGP and MRTd for monitoring, research, and statistics collection purposes. In this release of MRT, route_btoa supports the parsing of BGP, BGP+ and RIPng packets.

Route_btoa includes a Perl version of the program. In general, the compiled version is probably more robust and up-to-date than the Perl code.

## Synopsis

**route_btoa [-m] [-i input_binary_file]**

## Options

**-i binary_data_in_file**

Read routes from this file binary MRT file. Using a file name of 'stdin' will read input from standard in.

**-m**

Create machine-parseable output.

## Description

The following command writes a formatted, ASCII description of BGP4 update packets from a peering session with the NSFNET backbone to standard output:

**sbgp -bo stdout | route_btoa -i stdin**

Below is an example of the output produced by route_btoa. Most of the fields should be self-explanatory.

```
> /statistics/bin/route_btoa2 -i /cache/mae-east/bgp.980114.21:30
TIME: 01/14/98 21:30:00
TYPE: BGP/UPDATE
TO: AS2885   192.41.177.169
FROM: 4.0.0.10 AS1
ASPATH: 1
ORIGIN: IGP
NEXT_HOP: 192.41.177.2
MULTIEXIT: 1546
ANNOUNCE:
  140.249.0.0

TIME: 01/14/98 21:30:01
TYPE: BGP/UPDATE
TO: AS2885   192.41.177.169
FROM: 144.228.107.1 AS1239
ASPATH: 1239 6453 5769
ORIGIN: IGP
NEXT_HOP: 192.41.177.241
MULTIEXIT: 91
ANNOUNCE:
```

Route_BtoA also supports the generation of machine-readable output. This mode generates output that is easily parsed by awk or Perl scripts for statistics calculations. Note that "**-m**" mode does not preserve information about packet boundaries. The format for each line of the machine-readable output for BGP4 and BGP4+ packets is:

**Protocol** | **Time** | **Type** | **PeerIP** | **PeerAS** | **Prefix | <update dependant information>**

Where **protocol** is BGP, or BGP4. The **time** is number of seconds since epoch when the packet was recorded. The **type** is A for announcement, or W for withdrawal. **PeerIP** and **PeerAS** are the IP address and AS number of the BGP peer from which we received the update. **Prefix** is the route prefix described in the update.

For BGP announcements, update-dependant information contains:

**ASPATH** | **Origin** | **NextHop** | **Local_Pref** | **MED | Community**

Where **ASPATH** is the autonomous system path of the update. **Origin** is IGP, EGP, or Unknown. And **local_pref**, **MED** and **Community** are as the names imply. Below is an example of route_btoa machine output of MRTd-collected BGP packets:

```
BGP|884831400|A|4.0.0.10|1|140.249.0.0/16|1|IGP|192.41.177.2|0|1546
BGP|884831401|A|144.228.107.1|1239|205.113.0.0/16|1239 6453
      5769|IGP|192.41.177.241|0|91
BGP|884831402|W|204.70.7.53|3561|198.163.111.0/24
BGP|884831402|W|204.70.7.53|3561|199.212.219.0/24
BGP|884831402|W|204.70.7.53|3561|199.235.123.0/24
BGP|884831402|W|204.70.7.53|3561|204.112.101.0/24
BGP|884831402|W|204.70.7.53|3561|204.112.232.0/24
BGP|884831402|W|204.70.7.53|3561|205.189.8.0/24
BGP|884831402|W|204.70.7.53|3561|205.211.8.0/24
```

# 7. ROUTE_ATOB

Route_AtoB converts ASCII descriptions of MRT messages to binary. By default, the program writes binary MRT message streams or files to standard out.

Route_AtoB includes a perl version of the program. In general, the compiled version is probably more robust and up-to-date than the perl code.

## Synopsis

**route_atob -i ASCII_input_file**

## Options

### -i ascii_input_file

Read ASCII descriptions of MRT messages from this file. Using a file name of 'stdin' will read input from standard in.

## Description

The following command sequence will inject the routes described in in the input file into the BGP peering session with enss131.

**route_atob -i /tmp/input | sbgp -i stdin enss131 AS690**

See the output of route_btoa for the grammer required by route_atob.

# 8. Data Distiller

Data Distiller generates summary reports on the number of routing messages seen at one or more peering points. Data are gathered by passive (i.e. listening only) participation in the BGP peering session.

## Synopsis

**datadstl [-f config_file] [-p uii_port] [-v] [-d]**

## Options

**-d**

Run Data Distiller as a daemon. The process detaches from its controlling tty, closes inherited file descriptors, etc. This option overrides the **-v** option.

**-f configuration_file**

Read the specified configuration file. If Data Distiller is not running in daemon mode, the default is datadstl.conf in the working directory. If running in daemon mode, the default is datadstl.conf in the root directory.

**-p**

Specifies the port on which to accept UII telnet connections. (Default: 5680)

**-v**

Log to stdout. If this option is specified and a log filename is included in the config file, log messages will be sent to stdout until the appropriate line in the config file is read. Note that the **-d** option overrides the **-v** option. Running Data Distiller as a daemon requires closing all inherited file descriptors, including stdout/stderr.

## Description

Data Distiller reads raw MRTd data files and cooks them into a variety of periodic reports. These reports are disseminated to the IPMA Java clients using either a web (HTTP) server, or the Dolphin server.

Data may be collected either by Data Distiller, or by a separate MRTd process. The former is the simplest way of running Data Distiller. However, in certain situations, running separate MRTd processes may be desired. You may, for example, wish to limit the load on the machine at the peering point.

If Data Distiller is responsible for data collection, it will need access to low numbered ports so that it can establish a BGP peering session. On most systems, this means root access.

If MRTd is collectiong the data, Data Distiller must be able to access the filesystem to which MRTd is dumping the data. Distributed file systems (such as NFS) work well with Data Distiller. If the data will be in AFS, please be sure to take heed of token expiration issues.

## Operations

Data Distiller periodically updates its state from the MRTd raw data files. It then generates

cooked data. This data can either be stored using the file system, or published to the Dolphin data dissemination server. Any exceptional events (such as the server disconnecting) are logged. Data Distiller can also be configured to log more mundane events, such as processing of the raw data files, or the action of outputting each cooked data file.

### Telnet Interface

Data Distiller uses the MRT UII facility for its telnet interface, and supports standard MRT commands for remote monitoring and administration. The telnet interface can be disabled through the configuration file.  Note that Data Distiller does not look in /etc/services for its UII port number.

Data Distiller also supports remote configuration. Any command that may appear in a configuration file may also be issued via the telnet interface.

### Configuration

Although Data Distiller can be configured over a telnet connection, it is easier, and more reliable, to do so via the configuration file. Below is a sample.

```
! ASExplorer, FlapGraph, and FlapTableDaily are the currently supported cooked
! data types. Each corresponds to one Java visualization tool.
!
! A command that begin with any of these applies only
! to that data type. All other commands are "global" -- they affect the
! operation of Data Distiller generally.
!
! ------------------------------------------------------------------------
!
!
! Specify logging.
!   format: log <level> [logpath max-length]
!   log normal messages to /tmp/latest.log. don't exceed 100K for logfile
debug norm /tmp/latest.log 100000
!   also log trace-level messages
debug trace
!
! ------------------------------------------------------------------------
!
!
! UII Configuration
!    which port to listen on?
!      format: uii_port <port #>
!      listen on port 5690
uii_port 5690
!    what's the password for access?
!      format: password <string>
!      password is CHANGEME
password CHANGEME
!
! ------------------------------------------------------------------------
!
!
```

```
! Data Capture
!
! peering -- who do we capture BGP traffic from?
!   who are we?
!     format: router bgp <local as number>
!     we are AS -5000
router bgp -5000
!   who do we peer with, and where?
!     format: neighbor <ip addr> remote-as <as number>
!     peer with AS 0 at 192.168.1.1
neighbor 192.168.1.1 remote-as 0
!     and AS -1000 at 192.168.5.5
neighbor 192.168.5.5 remote-as -1000
!
! storage -- where do we put the captures, and how often?
!   format: dump bgp updates <capture path>/<peering point>/bgp.%y%m%d.%H:%M
<interval>
!   store the captures in /bgpdata/mae-east, and dump the data every 15
minutes
!   if you change the format specifier for the date, you'll need to change
!   parsename in bgp_db.c
dump bgp updates /bgpdata/mae-east/bgp.%Y%m%d.%H:%M 15m
!
! ----------------------------------------------------------------------
!
!
! "Global" Configuration (affects all cooked data types)
!
! Tell Data Distiller where to find the datafiles.
!   format: database_directory <capture path>
!   our files are in /bgpdata.
!   (this is the same as capture path in "dump bgp updates".)
database_directory /bgpdata
!
! The data files that we read have their starting time as part of the
filename.
! Data Distiller needs to know whether the time is in UTC or localtime.
!   format: filename_timezone_local {0|1}
!   filenames are in localtime
filename_timezone_local 1
!
! How much time is covered by each data file?
!   format: file_timespan <interval (in seconds)>
!   This is the same interval as in "dump bgp updates", except that this must
!   be in seconds.
file_timespan 900
!
! If Data Distiller is publishing data frequently (as it is in this sample
! conifguration), then a significant amount of time will be spent reading
! directory information. This can be avoided by using the directory cache.
! Here we indicate how long the cache is considered valid (a value of 0 will,
! of course, effectively disable the cache).
! Be careful not to set this value to more than the bucket size for the
! FlapGraph/FlapTableDaily data (15 minutes), because you may miss updates if
! you do.
!   format: dircache_time <time (in seconds)>
!   invalidate the directory cache info after 5 minutes
dircache_time 300
!
! ----------------------------------------------------------------------
```

```
!
!
! ASExplorer Configuration
!
! In our installation, the BGP data files are placed in directories according
to
! the peering point at which they were collected. But the ASExplorer tool
! would like to know which probe machine collected the data. We implement the
! mapping with these configuration commands. This is rather "cosmetic", and
! optional.
!   format:  ASExplorer probe_name <directory> <probe machine>
!   files in mae-east directory contain data collected by host rs2.mae-
east.rsng.net
ASExplorer probe_name mae-east rs2.mae-east.rsng.net
!   files in mae-west have data from rs2.mae-west.rsng.net
ASExplorer probe_name mae-west rs2.mae-west.rsng.net
!   files in aads have data from rs2.mae-west.rsng.net
ASExplorer probe_name aads rs2.aads.rsng.net
!
! How often should ASExplorer cooked data be updated?
!   format: ASExplorer update_interval <interval (in seconds)>
!   update ASExplorer data every 60 seconds
ASExplorer update_interval 60
! ASExplorer generates summary information for a sliding window. Here we
specify
! the window size.
!   format: ASExplorer report_time_range <interval (in seconds)>
!   gather ASExplorer data over the last 90 minutes
ASExplorer report_time_range 4500
!
! Where do we put the cooked data? There are two possibilities.
! We can either save the data in the filesystem, or we can feed it to
! the Dolphin server. Examples follow.
!
!   Output Method?
!   format: ASExplorer output_method {dolphin|files}
ASExplorer output_method files
!   Method specific configuration: files
!     Where to put the output files?
!     format: ASExplorer basedir <path>
!     place the files in /web/stats. as /web implies, this should
!     be a directory accessible to your webserver.
ASExplorer basedir /web/stats
!   Method specific configuration: dolphin
!     What's the hostname of the Dolphin server?
!     format: ASExplorer salamander <hostname>
! ASExplorer salamander nowhere.nil
!     What port is Dolphin listening on?
!     format: ASExplorer port <port #>
! ASExplorer port 8899
!
! ------------------------------------------------------------------------
!
!
! FlapGraph Configuration
!
! These commands are analagous to the ones for ASExplorer. See the ASExplorer
! example above for details.
!
FlapGraph update_interval 90
```

```
FlapGraph output_method files
FlapGraph basedir /web/stats
!
!
! FlapTableDaily Configuration
!
FlapTableDaily update_interval 105
FlapTableDaily output_method files
FlapTableDaily basedir /web/stats
!
! ------------------------------------------------------------------------
!
!
! Miscellaneous Configuration
!
! FlapGraph and FlapTableDaily data contain summary information about
acitivity
! since midnight. This can be done based on localtime, or UTC. We recommend
! using UTC. (Note: this option is not independently configurable. The command
! can be issued referencing either FlapGraph or FlapTableDaily, but will
affect
! both.) This option is primarily for backward compatibility with some
internal
! tools. You should use UTC.
!   format: {FlapGraph|FlapTableDaily} use_localtime {0|1}
!   publish FlapGraph (and FlapTableDaily) data in UTC
FlapGraph use_localtime 0
!   equivalenty, we could have used FlapTableDaily use_localtime 0
!
```

### Configuring Client Tools

In <basedir>, create a file called .conf. It should have the following information:

```
Transport=HTTP
ToolListInternicURL=http://www.merit.edu/ipma/ipmaconf/
ASExplorer.RefreshInterval=<"ASExplorer update_interval" value>
FlapGraph.RefreshInterval=<"FlapGraph update_interval" value>
FlapTable.RefreshInterval=<"FlapTable update_interval" value>
```

The first two lines should be copied exactly. In the last three lines, substitute the apporpriate values from the Data Distiller configuration file.

### File Directories

The various directories might be arranged as follows: