Chapter 3
Customizing Your Build Configuration

3.1 Overview

This release of GateD uses GNU autoconf to detect platform-specific configuration options. You do not need to install autoconf to build GateD. By default, GateD is compiled with all protocols available in the license under which it was distributed.

This section documents how the build process may be configured to set the protocols and features compiled in the GateD binary and change various compile-time parameters.

Several binaries may be produced by a compile of GateD, including standalone utilities such as 'ripquery' and 'gdc'. These binaries will be present in a directory off of 'src' named the same. The 'gated' binary is produced in src/gated.

Most build configuration is performed by passing command-line flags to the 'configure' script. For a complete list of options, use:

```
./configure --help
```

3.2 Build Tools

The autoconf system provides some built-in flags for setting the tools to be used for compilation. The GNU web site, www.gnu.org, is the most complete source of documentation for these options. A few important ones are documented here.

3.2.1 C Compiler

The path to the C compiler to be used may be set with either the **`--with-cc`** flag or by setting the $CC environment variable. For example, to build GateD using cc, use:

```
./configure --with-cc=cc
```

3.2.2 Parser Generator

GateD uses a yacc-based parser for parsing the configuration file. This requires a tool such as yacc or bison to generate the C source code from a set of rules. The path to this tool may be set by using the **`--with-yacc flag`** or by setting the $YACC environment variable. For example, to use bison, use:

```
./configure --with-yacc=bison
```

3.2.3 Lex Scanner

GateD requires a lexer generator, which generates a lexer C source file from a set of rules. To set the path to this tool, use the **`--with-lex flag`** or set the $LEX environment variable. For example, to use flex, use:

```
./configure --with-lex=flex
```

3.2.4 Installation Paths

See the GNU autoconf manual for a complete list of options concerning the installation paths. Compilation of GateD will typically result in several standalone binaries, which may be installed in different locations. The GateD binary is installed in the 'sbin' directory specified by 'configure'.

3.3 Protocols

The 'configure' script may be used to configure a set of protocols to be built in this GateD binary. Most protocol modules have **`--enable`** and **`--disable`** flags, which may be used to enable or disable the protocol, respectively. The **`--disable-all`** flag is also useful because it allows the user to disable all protocols and then enable a small subset. For example, to build a GateD binary with only RIP, use:

```
./configure --disable-all --enable-rip
```

Dependencies are automatically resolved by 'configure'. Enabling the BGP module, for example, also enables the ASPATHS module.

3.4 Standard Features

Some features of GateD can be enabled or disabled at compile time. This is in addition to the protocols themselves; the features documented here are optionally enabled to support some particular functionality or different mode of operation.

All of the features defined here can be enabled or disabled by passing flags to the 'configure' script, which is run before building GateD.

3.5 Additional Features

In addition to this standard set of features, there are some other parameters that can be manipulated to fine-tune the protocols' compile-time configuration. These parameters are documented here. All parameters are disabled by default.

Feature: GII_DEBUG_MENU

Flag: --enable-developer

The GateD Interactive Interface (GII) contains some protocol debugging functions. For example, one function allows a neighbor's Router-LSA to be sequence-wrapped and reflooded. These functions are available in the 'debug' submenu when GII_DEBUG_MENU is enabled.

In addition to enabling GII_DEBUG_MENU, the --enable-developer flag also turns on some strict compiler warning flags when certain types of compilers are used (gcc is one of them).

Feature: NOSPF_ALWAYS_CYCLE_MEMBERSHIP

Flag: --enable-ospf_dropfirst

Some versions of BSD exhibit a bug that prevents a multicast group from being dropped on a logical interface when that interface has been deleted. This option forces GateD to always drop and join when attempting to join a multicast group.

Feature: FEATURE_NOSPF_ALTQ_TE

Flag: --enable-ospf_altq_te

In the new_ospf protocol module there exists code to support learning bandwidth information from the /dev/cbq device in a BSD kernel with ALTQ patches. This code requires that the ALTQ header files exist in /usr/include/altq.

Feature: FEATURE_NOSPF_FAST_SPF

Flag: --enable-ospf_fast_spf

This feature expands the size of the **`vertex_t`** structure (the structure used to represent a Link State Advertisement) in order to gain speed in the SPF. An extra pointer is added, increasing the size by one machine word (typically 32 bits). The result is an approximate twenty-five percent reduction of computation time required by the SPF. The pointer is used to keep track of a candidate list entry for a vertex.

Feature: FEATURE_INTERFACE_AGING

Flag: --enable-interface_aging

Interface aging is a legacy behavior of GateD. In older code bases, unless the 'passive' option was specified on an interface, an interface that did not receive routing protocol traffic for a certain period of time was downed. This feature is now disabled unless explicitly requested here.