Using GateD V.9.3.2 with VxWorks

1.1 Introduction

This document describes the build and operation of the NextHop GateD software with the VxWorks operating system from Wind River Systems, Inc. Using GateD with a real time operating system (RTOS) requires special consideration in the configuration and build processes due to the typical UNIX daemon model for which GateD was originally designed. When these special considerations are addressed, GateD functions normally as a single task in the VxWorks RTOS.

It is assumed that the reader is familiar with the basic layout of the GateD source code and architecture.

2.1 Supported Versions

GateD has been compiled and tested under the VxWorks environment shown in Table 1.

Component	Type and Version
Cross Development OS	Solaris 8 SPARC
VxWorks Kernel	5.4.2
Board Support Package (BSP)	MCP750 1.2/3
Compiler	ccppc cygnus-2.7.2-960126
Debugger	gdbppc 4.17
Tornado	2.0.2

Table 1

While the VxWorks operating environment can be configured in Tornado, it is strongly recommended that GateD be built using the manual tools supplied with the distribution rather than with the Project facility. The output of the manual build process can then be used to link a final image with Tornado Project. Using the manual build process allows the normal autoconfiguration, dependency tracking, and build structure to be maintained.

3.1 Configuration and Build

3.1.1 Overview

GateD uses the GNU autoconf and automake packages for build configuration. See the GNU web site, www.gnu.org, for more information on these packages. When certain parameters are tuned, the autoconf/automake process is compatible with the VxWorks build environment. The process of creating a GateD image that can run as a task in the VxWorks kernel can be summarized as follows:

1. The **`configure`** script is run in the GateD build directory, which outputs the Makefiles and files that contain build-specific information.
2. **`make depend`** is run to create dependencies in the Makefiles.
3. **`make`** is run, which descends into the protocol subdirectories and spawns the compiler to compile the module source libraries.
4. A relocatable object file is created (using the **`-r`** flag) in the **`src/gated`** directory.
5. The kernel is configured to include any extra services that GateD uses.
6. The image is linked with the kernel, VxWorks, in the BSP config directory.

A convenient way to build GateD for multiple architectures while leaving the source tree intact is to make a directory *root*/**object**, where *root* is the base GateD source directory. The following example:

```
makedir -p obj/machfoo
cd obj/machfoo
../../configure flags && make depend && make
```

makes a set of binaries in **`obj/machfoo`**. Other directories in **`obj`** can be created while leaving the source directories untouched. See the README file in the base installation for more details.

3.1.2 Kernel Configuration

GateD relies on several services in the kernel which are not enabled in the default distribution of VxWorks. These services are shown in Table 2. They must be enabled in the kernel before linking with GateD.

Parameter	Purpose
`INCLUDE_POSIX_TIMERS`	POSIX timers for itimer module
`INCLUDE_POSIX_SIGNALS`	POSIX queued signals for several operations, for example, **`SIGHUP`**
`INCLUDE_ROUTE_SOCK`	BSD routing socket for manipulation of the routing table

Table 2

3.1.3 GateD Build Configuration

3.1.3.1 Configure Flags

The configure shell script is used to determine build-time parameters before running make and make depend. The modular structure of GateD allows for protocols to be excluded or included from the build. Several additional parameters, such as debugging menus, SPF algorithms, and the sizes of certain core data structures can be adjusted using this script as well. See “Building GateD” on page 4 for a complete description of all available parameters. Only the parameters relating to building in the VxWorks environment are discussed here.

The following flags must be given to configure when building for VxWorks. Other flags are optional:

```
--enable-vxworks --disable-lock-pid-file --disable-lock-trace-file
--disable-lock-dump-file --disable-write-pid-file --disable-main
--disable-all --enable-bgp --enable-ospf --enable-mpbgp --enable-rip
--enable-gii --enable-isis
```

This prevents several files from being locked because this is an unnecessary action in an RTOS environment. The **`main`** function will not be compiled; this is necessary with some compilers that output special code when **`main`** is used. The protocol flags here enable the set of protocols supported in VxWorks for this release and the actual protocols enabled may depend on your distribution. Finally, the VxWorks specific build parameters are enabled.

3.1.3.2 Toolchain Flags

Several environment variables can be set to indicate flags that should be used with the compiler, **`ccppc`**.

The following flags must be included in the settings of the environment variables shown in Table 3. These flags must be included; the actual set of flags used can vary according to your requirements (for example, compiler optimization flags can be used).

Environment Variable	Value
$CFLAGS	**-DCPU=PPC604 - ansi -nostdinc -g -fno -builtin -fvolatile** **-DRW _MULTI_THREAD - D_REENTRANT -I$WIND_BASE/target/h**
$CC	**ccppc**
$CPP	**ccppc -E**
$LDFLAGS	**-r**
$CCPPFLAGS	**-nostdinc -I$WIND-BASE/target/h -DCPU=PPC604**
$WIND_BASE	base of Tornado installation

Table 3

3.1.4 Building GateD

After configuring the parameters described above, the following commands can be issued to build GateD. The output of this process is a relocatable image in *root*/**obj/gated**, which can be linked with the VxWorks kernel.

```
configure flags
make depend
make
```

4.1 Running GateD as a VxWorks Task

GateD has an internal tasking structure that typically runs as a standard UNIX daemon. It has been ported to function as a VxWorks task.

4.1.1 Starting GateD

The **vxw_gated_begin** function in **gatedcompat/vxw_ctl.c** is intended to serve as the startup function for the GateD task. For example, GateD is started from the controller (see “The GateD Controller Task” on page 5 of this document) as follows:

```
rv = taskSpawn(VXWC_GATED_NAME, VXWC_GATED_PRIORITY, VXWC_GATED_OPTIONS,
VXWC_GATED_STAC_SIZ, VXWC_GATED_BEGIN_FUNC, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0);
#define VXWC_GATED_NAME “gated”
#define VXWC_GATED_PRIORITY 46
#define VXWC_GATED_OPTIONS(VX_FP_TASK)
#define VXWC_GATED_ID 1
#define VXWC_GATED_STAC_SIZ 20000
#define VXWC_GATED_BEGIN_FUNC vxw_gated_begin
```

The prototype for this function is as follows:

static int vxw_gated_begin(*int*, *int*, *int*, *int*, *int*, *int*, *int*, *int*, *int*, *int***);**

The values for priority and stack size are the default values from **sp** and may need to be changed to suit your environment. The **VX_FP_TASK** flag is required for some floating point operations that may not be supported in all distributions.

5.1 The GateD Controller Task

An example GateD controller task is included in the file **gatedcompat/vxw_ctl.c** and is compiled by default. This task can be started by calling the function, **gated_control_startup**.

This task is provided for use in laboratory testing as a convenient way of starting and stopping GateD and is not recommended for production use as no authentication procedures are provided. The controller task, by default, answers TCP requests on TCP port 2112. The command descriptions are shown in Table 4.

Command	Description
set config file: *file*	Sets the path to the GateD configuration file
reconfig	Sends a SIGHUP to GateD, causing it to re-read the configuration file
running	Prints a message showing whether GateD is running
start	Starts the GateD task
stop	Stops the GateD task
show routes	Prints the VxWorks routing table
dump	Sends a SIGINT to GateD, causing a dump file to be written (not supported at this time)
reboot	Reboots the VxWorks kernel

Table 4

6.1 Known Issues

This section lists known issues in the release of GateD when it is used with VxWorks.

6.1.1 Restarting the GateD Task

Stopping and restarting GateD is not supported. The typical UNIX process model performs cleanup on behalf of the process when it exits. This does not occur under the VxWorks “tasking” scheme. Therefore, in order for GateD to run properly, it must be started only once in a boot cycle. At termination (for example, upon receipt of a TERM signal), GateD will free its memory but will not support restarting unless the kernel is reloaded first.

6.1.2 Dump Files

The GateD dump facility is supported, but because VxWorks does not implement the traditional **`fork(2)`** system call, dumping takes place in line with the main task and can delay other functions. It should be used as a debugging aid only.

6.1.3 GII Authentication

The GateD Interactive Interface (GII) relies on user authentication from a traditional UNIX password file. Because VxWorks does not support the concept of “users” and does not have such a database, GII defaults to no authentication.

6.1.4 Task Priority

The VxWorks Network Programmer’s Guide suggests that a task depending on networking facilities should be assigned a priority better than the network task. The default GateD priority is 46, which is less than the network task priority of 50. Be careful not to change this value such that the network task is starved.

6.1.5 Name Resolution

Name resolution of hostnames in the configuration file is not supported.

6.1.6 Default Page Size

A default page size of 4096 bytes is used if no other setting is given. The **`--with-page_size=`**`bytes` flag can be given to configure or change this value.

6.1.7 ISIS and Physical Layer Access

The ISIS implementation utilizes the VxWorks MUX API for sending and receiving packets. Since the MuxTk versions of these functions do not allow access to the physical layer header of received packets, only END (Enhanced Network Driver) style network drivers are supported with ISIS.

6.1.8 Network Memory Pool Configuration

It may be necessary to adjust the amount of memory allocated by the network stack of the operating system. If GateD is used in an environment that requires heavy use of I/O (for example, many OSPF neighbors), the number of clusters may need to be increased. Consult the operating system manual for details.