Configuring and running DTN2 and the HBSD external daemons

Configuration

In this section, we'll prepare an appropriate dtn.conf file for our instance of dtnd.

Directories

First, you need to choose a directory that will hold the files dtnd needs. For the purposes of this tutorial, we'll assume your username is amir and that you want dtnd to write into /home/amir/hbsd-dtn2-conf.

Make the /home/amir/hbsd-dtn2-conf directory, then copy the example dtn.conf file from DTN2/daemon/dtn.conf to /home/amir/hbsd-dtn2-conf/dtn.conf.

Editing dtn.conf

Edit the dtn.conf file.

We'll use the simplest database to configure, BerkeleyDB.
Leave the storage set type berkeleydb line alone.

Set the payloaddir to /home/amir/hbsd-dtn2-conf/bundles and set the dbdir to /home/amir/hbsd-dtn2-conf/db.

The dbdir directory is where persistent objects that DTN2 uses to manage its internal state live. This database will remain fairly small. It needs to be preserved between invocations of the DTN2 daemon, because the data in it lets DTN2 know the status of registrations with clients, and bundles which are in flight in the network.

One of the "disruptions" that a DTN is tolerant of is the failure of a node. Using the data in the database, a new DTN2 daemon can carry on the work the previous one was doing. But note that if the contents of the database are lost, it is as if this node never existed, and there will be repercussions in the DTN (like retransmissions of bundles) as a result.

The payload directory is where bundles which are in-flight live. In general, the bundles in this directory are ones which this DTN2 instance has custody of. That means that if they disappear out of this directory, they will be lost forever, since there are no other authoritative copies of the bundle in the network.

DTN2 and the DTN protocols are designed to maintain the files in this directory without leaving old ones lying around. If you see an old file there, it's likely due to a bug.

On a server dedicated to running DTN2, your payload directory should have access to the largest partition on your server. As an analogy, on a mail server the mail spool is usually on /var, and has access to the majority of

the extra disk space on a machine. The same should be true for the payload directory.

Router EID:

You don't need to make any changes in the local_eid, but scroll down and take a look at it anyway. In the defualt configuration, it looks like this: route local_eid "dtn://[info hostname].dtn". Note the square brackets. To understand what's happening here, you need to know that this configuration file is not simply a file of settings for dtnd, but an actual TCL script.

It is interpreted by the TCL interpreter that is built in to dtnd. In TCL, square brackets mean "interpret this first, and replace the brackets with the result". So before dtnd sees a route local_eid ... command, TCL processes the info hostname command. As you might be able to guess by now, the entire line results in the local_eid being set to a custom string based on your hostname.

This is the only place in this particular configuration file where we will use TCL features like the square brackets. However, you should remember that all the power of TCL is available to you as you write your own configuration files, should you desire it. For an example of this, see the example showing how to rotate logs from within dtnd.

So the local EID is set, but what is it? From the Architecture section, you should recall that nodes in the DTN are identified by EIDs. The local EID is simply the way our DTN node will refer to itself. Without this setting, it would not recognize bundles intended for it, and might just pass them on ad infinitum! People who have configured Internet email systems like Postfix or Sendmail might recognize this setting, as mail routers need to know their own name for the same reason.

And EID is a string that is used throughout the DTN to identify an endpoint. That leaves you considerable latitude on how to set it. We recommend for now that you set it according to the machine name where the DTN node is running, as the example does.

At this time, one dtnd can have only one local EID. This limitation might disappear in future versions if it proves to be a problem.

Interfaces:

Take a look at the interface add commands a few lines down. The command interface add tcp0 tcp adds a TCP convergence layer named tcp0 to the server. This means that the daemon will start listening on port 4556 for incoming connections from other servers in the DTN.

Usually DTN nodes that are communicating via TCP listen on port 4556, based on the IANA port assignment. If you have another application on the same server listening on port 4556, you'll need to choose a different port for your dtnd to use.

The interface add command can take convergence layer specific arguments to customize the behavior of the interface. For the purposes of this tutorial, we will be using the defaults. For more information on these arguments see the reference pages on the various convergence layers.

In some security contexts, and with some virtual interface configurations, it is desirable to have dtnd listen only on a certain interface, for instance "listen to internal connections only". You can do this by adding local_addr=desired-ip to the end of the interface add command. The default local_addr is 0.0.0.0, meaning "listen on all interfaces". For this tutorial, we will allow dtnd to listen on all interfaces, so leave the interface add command as it is.

Note that we are adding an interface of type TCP. There are other ways that DTN nodes can communicate, including via UDP, raw ethernet frames, and across a filesystem (where some external activity, like a messenger with a USB keychain drive, is responsible for moving the files). A server that is using multiple convergence layers has other interface add lines in the configuration file.

In fact, below the interface add line for the TCP convergence layer is one for the UDP covergence layer. Because we are setting up a standalone server in this tutorial, neither udp0 nor tcp0 will end up getting used to move bundles at this point.

Links and Routes:

HBSD requires that the DTN2 configuration file include the following line:

param set early_deletion false

If this option is not set then DTN2 may delete bundles after they have been initially sent, preventing HBSD external router from replicating bundles.

You also need to make certain that DTN2 is configured to use an external router:

route set type external

HBSD also assumes that DTN2 is configured to open and close links as nodes come in and out of contact.

This may include using a discovery protocol.

Initializing the DTN2 Database

Before dtnd can use the database to keep track of runtime state, it needs to initialize it.
This is a simple matter of starting the daemon once with the --init-db argument.
You'll also want to give it the location of the configuration file, so that dtnd can find the correct database directory. Put that together and you have:

```
$ ./daemon/dtnd -c /home/amir/hbsd-dtn2-conf/dtn.conf --init-db
```

Running the DTN2 Daemon

For now, you start dtnd from the commandline, like any other Unix command. By default, the TCL interpreter takes input from standard in, so you will need to leave it running in the foreground. While you are learning the ins and outs of dtnd, it is best to run it this way, so that you can interact with it.
Use the -c argument to tell it where to find the configuration file. For information on other arguments you can give dtnd, see the dtnd man page.

Once you start dtnd with a command like DTN2/daemon/dtnd -c /home/amir/hbsd-dtn2-conf/dtn.conf, it will put out some messages, then give you the dtn% prompt. This means the server is up and running, awaiting commands from you. Things you type at the prompt are interpreted by the same TCL interpreter that just parsed the configuration file. In a way, you can consider the configuration file a list of commands that will be issued for you each time you start the server.

Here's an example of starting up the server:

```
$ daemon/dtnd -c /home/amir/hbsd-dtn2-conf/dtn.conf
```

You get a bit more info if you start up with the '-l info argument':

Use the online help system to learn what you can do from here. Type help. For help on a specific command, type help command.

Detailed DTN2 configuration file:

```
# dtn.conf

log /dtnd info "dtnd parsing configuration..."

########################################
# Daemon Console Configuration
########################################

# console set addr <port>
# console set port <port>
# Settings for the socket based console protocol.
# (this interprets user commands)

console set addr 127.0.0.1
console set port 5050

# console set prompt <prompt>
# Set the prompt string.  Helps if running multiple dtnd's

set shorthostname [lindex [split [info hostname] .] 0]
```

```
console set prompt "$shorthostname dtn% "

########################################
# Storage Configuration
########################################

# storage set type [ berkeleydb | external | memorydb ]
# Set the storage system to be used

storage set type berkeleydb

# the following are for use with external data stores
#
# The server port to connect to (on localhost)
# Note that 62345 has no special significance -- chosen randomly

storage set server_port 62345

# The external data store schema location, which can be
# found by default in dtn2/oasys/storage/DS.xsd.

storage set schema /home/amir/hbsd-dtn2-conf/DS.xsd

# Do a runtime check for the standard locations for the persistent
# storage directory

set dbdir "/home/amir/hbsd-dtn2-conf"
foreach dir {/var/dtn /var/tmp/dtn} {
    if {[file isdirectory $dir]} {
        set dbdir $dir
        break
    }
}

if {$dbdir == ""} {
    puts stderr "Must create /var/dtn or /var/tmp/dtn storage directory"
    exit 1
}


# storage set payloaddir <dir>
# Set the directory to be used for bundle payload files

storage set payloaddir $dbdir/bundles

# storage set dbname <db>
# Set the database name (appended with .db as the filename in berkeley
```

```
# db, used as-is for SQL variants
#

storage set dbname     DTN

# storage set dbdir    <dir>
# When using berkeley db, set the directory to be used for the
# database files and the name of the files and error log.

storage set dbdir      $dbdir/db

########################################
# Routing configuration
########################################

# Set the algorithm used for dtn routing.
# route set type [static | flood | neighborhood | linkstate | external]

route set type external

# route local_eid <eid>
# Set the local administrative id of this node. The default just uses
# the internet hostname plus the appended string ".dtn" to make it
# clear that the hostname isn't really used for DNS lookups.

route local_eid "dtn://[info hostname].dtn"
#
# External router specific options
#
route set server_port 8001
route set hello_interval 10
route set schema "/home/amir/HBSD_DTN2/Config/router.xsd"


########################################
# TCP convergence layer configuration
########################################

# interface add [name] [CL]
# Add an input interface to listen on addr:port for incoming bundles
# from other tcp / udp convergence layers
#
# For IP-based interfaces, interfaces listen on INADDR_ANY port 4556
# by default. These can be overridden by using the local_addr and/or
# local_port arguments.

interface add tcp0 tcp
```

```
interface add udp0 udp

# link add <name> <nexthop> <type> <clayer> <args...>
# Add a link to a peer node.
# For IP-based links (tcp or udp), the nexthop should contain a DNS
# hostname or IP address, followed optionally by a : and a port. If
# the port is not specified, the default of 4556 is used.
#
# e.g.  link add link1 dtn.dtnrg.org ONDEMAND tcp
#       link add link2 dtn2.dtnrg.org:10000 ONDEMAND tcp


# route add <dest> <link|peer>
# Add a route to the given bundle endpoint id pattern <dest> using the
# specified link name or peer endpoint.
# e.g. route add dtn://host.domain/* tcp0


########################################
# Service discovery
########################################

# discovery add <name> <af> <opts...>
# discovery announce <cl_name> <discovery_name> <cl_type> <opts...>

# Add a local neighborhood discovery module

discovery add discovery_bonjour2 ip port=4557
discovery announce announce-tcp0 discovery_bonjour2 tcp interval=5

# e.g. discovery add discovery_bonjour bonjour

########################################
# Parameter Tuning
########################################

#
# Set the size threshold for the daemon so any bundles smaller than this
# size maintain a shadow copy in memory to minimize disk accesses.
#
# param set payload_mem_threshold 16384

#
# Test option to keep all bundle files in the filesystem, even after the
# bundle is no longer present at the daemon.
#
# param set payload_test_no_remove true
```

```
#
# Set the size for which the tcp convergence layer sends partial reception
# acknowledgements. Used with reactive fragmentation
#
# param set tcpcl_partial_ack_len 4096

# Set if bundles are automatically deleted after transmission

param set early_deletion false

# (others exist but are not fully represented here)

log /dtnd info "dtnd configuration parsing complete"

## emacs settings to use tcl-mode by default
## Local Variables: ***
## mode:tcl ***
## End: ***
```

Detailed HBSD External router configuration file:

```
# Example HBSD configuration file.
# Author: Amir Krifa
# Email: krifa.amir@gmail.com

# To define this external router's endpoint, which is typically used to
# receive control messages from peer instances of the router. An example
# is meta data exchanged between the routers. DTN special-cases endpoints
# that begin with "ext.rtr/".
routerEndpoint=ext.rtr/HBSD

# Defines the multicast group the HBSD router is to join for communicating
# with the DTN daemon.
multicastGroup=224.0.0.2

# Defines the multicast port used by the DTN daemon.
multicastPort=8001

# The DTN daemon, dtnd, listens for messages from HBSD on a multicast
# socket. When HBSD sends requests to dtnd it does not want other
# DTN daemons or instances of HBSD on other systems to receive the
# multicast packets; the scope should be limited to the local host.
# This should be accomplished by setting the TTL for the socket to 0,
# but this doesn't always seem to work. Another approach is to use a
# datagram socket bound to the loopback interface. This doesn't work
# on other systems. The following defines which behavior to use. If
```

```
# set to true use a multicast source socket with a TTL of 0, otherwise
# use a source datagram socket bound to the loopback address.
multicastSends=false

# Address to bind to if multicastSends=false.
loopbackAddress=127.0.0.1

# File containing the XML schema definition for the messages exchanged
# with the DTN daemon. The command line takes precendence.
# Please consider putting an absolute path
xmlSchema=/user/akrifa/home/HBSD_DTN2/Config/router.xsd

# If set to false then the XML received from the daemon is not validated
# against the schema.
xmlValidate=true

# Allows for a user-specified logging class. The default is Console_Logging,
loggingClass=Console_Logging

# Logging configuration file. The format of the file is defined by the
# logging class being used. The command line takes precedence.
# Please consider putting an absolute path
logConfiguration=/user/akrifa/home/HBSD_DTN2/Log/LogFile

# Allows the logging level to be specified here rather than the command
# line. The command line takes precedence. If a logging level is not
# specified then the default is defined by the logger.
logLevel=6

# By default HBSD terminates when dtnd indicates that it is shutting
# down. Set the following to false to override the behavior.
terminateWithDTN=true

# Initial capacity of the map of all bundles on the system.
bundlesActiveCapacity=384

# Initial capacity of the map of all links.
linksHashCapacity=16

# Initial capacity of the map of all nodes.
nodesHashCapacity=32

#
# HBSD_Policy specific configuration parameters.
#

# Says whether to enable or disable the HBSD statistics based
```

```
# optimization or to just run the epidemic routing protocol
enableHbsdOptimization=false

# Selection of the optimization problem
# 0 means that the HBSD policy will try to manage both the buffer and links congestion in order to maximize the network average delivery rate.
# 1 means that the HBSD policy will try to manage both the buffer and links congestion in order to minimize the network average delivery delay.
hbsdOptimizePerformance=0

# The approximated number of nodes in the Network
numberOfNodesWithinTheNetwork=20

# If set to true  then the HBSD policy will use the online approximated number of nodes instead of the user provided one (numberNodes above)
useOnlineAproximatedNumberOfNodes=true

# The bin size. Could be approximated by the average nodes meeting time
binSize=100

# The length of the Bins table. Should be equal to the messages TTL (total time to live) devided by binSize
# Here we are supposing that all the generated messages have the same TTL value.
numberOfBins=360

# The maximum capacity of the MUM buffer, the one holding the message under monitoring.
mumBufferCapacity=50

# The maximum capacity of the MCH buffer, the one holding the message with complete history. (Note: keep this  buffer infinite if you are sure that
# your network will maintain the same behaviour during time otherwise choose the correct capacity in order to track the network dinamicity.
mchBufferCapacity=1000

# Specifies either to use the binSize as an expected average meeting time or to calculate online the average meeting time between nodes. Note
# that you should figure out an approximation of your network average meeting time in order to correctly trak the dinamicity of the network
# through the binSize
useBinSizeAsAvgMeetingTime=true

# Specifies wether to enable the MeDeHa interface or not
enableMeDeHaInterface=true
```

Need more help?

Please drop an email if you find any problem within the steps described above.
Amir.Krifa@sophia.inria.fr