DTNsuite

DTNsuite is a collection of DTN programs making use of the Abstraction Layer, plus the Abstraction Layer itself.

Being built on top of the Abstraction layer, DTNsuite applications are compatible with all the most important free bundle protocols implementations (DTN2, ION, IBR-DTN). IBR-DTN users are strongly suggested to rename the dtnd daemon of IBR-DTN as "ibrdtnd" to avoid any ambiguity with the "dtnd" daemon of DTN2. This would allow DTNsuite programs to recognize the active presence of IBR-DTN daemon correctly.

This document aims at providing the user with a brief overview of DTNsuite. For more information, see the specific documentation on the Abstraction Layer and application folders. Licenses of DTN suite components are reported on individual folders as well.

At present DTNsuite consists of:

- Abstraction Layer: a library that makes DTN applications independent of the specific APIs of BP implementations
- DTNperf: a powerful program for performance evaluation in DTN environments, plenty of features; its three modes (client, server, monitor) can also be used standalone (the client as a bundle generator, the monitor to collect status reports in .csv files.
- DTNproxy: a simple file transfer proxy from TCP/IP and DTN nodes and vice versa
- DTNbox: a complex peer-to-peer program for directory synchronization of DTN nodes, based on an external relational DB (SQLite); synchronizations can be bilateral or multilateral, with lots of features to match the peculiar characteristics of "challenged networks".

Project supervisor: Carlo Caini (carlo.caini@unibo.it).

Disclaimer: DTNsuite programs are distributed in the hope that they will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General Public License GPLv3 for more details.

1 BUILDING INSTRUCTIONS (FOR ALL DTNSUITE APPLICATIONS)

The DTNsuite package consists of a main directory, called "dtnsuite" with one subdirectory for each application, plus one for the Abstraction Layer. For example, the DTNperf code is in "dtnsuite/dtnperf". As all applications are based on the Abstraction Layer API, before compiling them it is necessary to compile the AL. For the user convenience it is possible to compile the AL and all applications at once (the drawback is that in this case it is more difficult to interpret possible error messages). Both ways are detailed below, starting from the latter, which is preferable for normal users.

Note that the compilation of DTNbox requires the presence of the SQLight package on the host. If not present, you must install it before compilation ("sudo apt-get update, sudo apt-get install libsqlite3-dev", in Ubuntu). Otherwise, you can comment the DTNbox compilation line in the /dtnsuite/Makefile.

1.1 INTEGRATED COMPILATION

It is possible to compile both AL and all DTNsuite applications in the right sequence with just one simple command. To this end, the Makefile in the "dtnsuite" directory calls the AL Makefile and the DTNsuite application Makefiles in sequential order.

The commands below must be entered from the "dtnsuite" directory. It is necessary to pass absolute paths to DTN2, ION and IBRDTN. The user can compile for one or more ION implementations as shown in the help:

for DTN2 only:

```
make DTN2_DIR=<DTN2_dir_absolute_path>
```

for ION only:

```
make ION_DIR=<ION_dir_absolute_path>
```

for IBR-DTN (>=1.0.1) only:

```
$ make IBRDTN_DIR=<IBRDTN_dir_absolute_path>
```

for all:

```
make DTN2_DIR=<DTN2_dir_absolute_path> ION_DIR=<ion_dir_absolute_path> IBRDTN_DIR=<IBRDTN_dir_absolute_path>
```

The AL libraries and the DTNsuite applications will have either the extension "_vION" or "_vDTN2" or _vIBRDTN, if compiled for one specific implementation, or no extension if for all.

It is also possible to compile for just two BP implementations, by passing only two paths in the command above.

Finally, the user needs to install the program in the system directory (/usr/local/bin) with the commands (with root permissions)

```
make install
ldconfig
```

Example:

```
<path to>/dtnsuite$ make DTN2_DIR=<path to>/sources/DTN2 ION_DIR=<path to>/sources/ion-open-source IBRDTN_DIR=<path to>/sources/ibrdtn
<path to>/dtnsuite$ sudo make install
<path to>/dtnsuite$ sudo make ldconfig
```

For recalling the help:

```
make
```

1.2 SEQUENTIAL COMPILATION

For a better control of the compilation process (e.g. if for whatever reasons it is necessary to change the AL or the specific application Makefiles), it is possible to compile AL and one or more DTNsuite applications sequentially, with independent commands.

1.2.1 Abstraction Layer

The Abstraction Layer (AL) must be compiled first; the AL compilation can be performed by means of the "make" command entered from the AL directory.

The possibilities are the same as those shown for the integrated compilation and will not reported here for brevity. Even in this case the AL will have either the extension "_vION" or "_vDTN2" or _vIBRDTN, if compiled for one specific implementation, or no extension if for all.

By default the AL library are static. If the default is overridden to dynamic (by modifying the AL Makefile), then the user needs to install the library in the system directory with the command (with root permissions):

```
make install
ldconfig
```

Example:

```
<path to>/dtnsuite/al_bp$ make DTN2_DIR=<path to>/sources/DTN2 ION_DIR=<path to>/sources/ion-open-source IBRDTN_DIR=<path to>/sources/ibrdtn
```

```
<path to>/dtnsuite/al_bp# sudo make install
<path to>/dtnsuite/al_bp# sudo ldconfig
```