Release Notes for DTNperf_3 June 2013

Release changes

DTNperf 3.x (DTNperf_3)
Authors: Michele Rodolfi, Anna d'Amico, Carlo Caini (project supervisor).
Although the aim is the same as previous versions, the project has been deeply redesigned and the code totally rewritten.
Three modes: client, server and monitor (new). They correspond to source, destination and "reply-to" dtn nodes. The monitor, in charge of collecting status reports in a .csv file, can be "internal" (on the source), or "external" (on a different node).
Server and external monitors can manage multiple instances of the client.
Three Tx modes: time, data and file. The first two are the same as before. The file mode is now much more robust as files segmented in multiple bundles can now be reassembled on the server even if received in disorder (but without losses).
Two congestion control methods: window based and rate based (new).
In the window based congestion control, bundle sent are now acknowledged by bundle ACKs sent by the server and no more by delivered status reports. In the rate based congestion control bundles are not acknowledged.
Support of both DTN2 and ION. The DTNperf application runs on top of a new "Abstraction Layer" (AL), which has the aim of decoupling the DTNperf application from the underlying BP implementation. The AL can be compiled for either DTN2, ION or both. DTNperf calls the API of the AL (or "grey" APIs), which in turns calls the API of either DTN2 or ION. If compiled for both, the choice of DTN2/ION APIs is made at run time, accordingly to the implementation that is actually running.
See the other sections (DTNperf compilation instructions, DTNperf general description and example of use) for further information.

DTNperf 2.x (DTNperf_2)
Authors: Piero Cornice, Marco Livini, Carlo Caini (project supervisor)
Window based congestion control added. With the –W parameter it is now possible to set the maximum amount of bundles in flight (sent but not acknowledged; "ACKs" are delivered status report). This is necessary to fill the pipe and to obtain accurate goodput evaluation.
Status reports of all machines are collected by the dtnperf client (the bundle source) in a .csv file, for later analysis on a spreadsheet.
Possibility to send bundles with dummy payload for either an interval time (time mode) or a specified amount of data (data mode).
Possibility to send a file, with file segmentation into multiple bundles of desired dimension (file mode).
The same version developed for GNU/Linux is ported with minimal changes on MAEMO OS for Nokia N900 smarthphones (by Francesco Baldassarri). It is fully interoperable with the version for Linux machines.
Bugs removed.

DTNperf 1.x
Author: Piero Cornice
First version of DTNperf, intended to be a DTN equivalent of the Iperf tool for DTN network (DTN2 implementation).
DTNperf client sends bundles of wanted dimension and dummy payload to dtnperf server, for a specified time interval.
Only one bundle in flight allowed ("in flight" means sent but not acknowledged yet; "ACKs"are delivered status report).

Instructions for the compilation of the Abstraction Layer and the DTNperf application.

Abstraction Layer
Before compiling DTNperf, it is necessary first to compile the Abstraction Layer (AL); the AL compilation can be performed by means of the the "make" command. There are three possibilities, depending on the BP implementation(s) you want to support:

DTN2 (≥2.9) only:
make DTN2_DIR=<dtn2_dir>

ION (≥3.1.2) only:
make ION_DIR=<ion_dir>

both:
make DTN2_DIR=<dtn2_dir> ION_DIR=<ion_dir>

Note that the AL will have either the extension "_vION" or "_vDTN2", if compiled for one specific implementation, or no extension if for both.

Example:
~/abstraction_layer_bundle_protocol$ make DTN2_DIR=/user/dtn
Then it is possible to install the library in the system directory with the command (with root permissions)

~/abstraction_layer_bundle_protocol$ make install

DTNperf application
Once the AL has been compiled and installed, DTNperf can be compiled in an analogous way:

DTN2 only:
make DTN2_DIR=<dtn2_dir> AL_BP_DIR=<al_bp_dir>

ION only:
make ION_DIR=<ion_dir> AL_BP_DIR=<al_bp_dir>

both:
make DTN2_DIR=<dtn2_dir> ION_DIR=<ion_dir> AL_BP_DIR=<al_bp_dir>

Example:
~/dtnperf_3$ make DTN2_DIR=/user/dtn AL_BP_DIR=~/abstraction_layer_bundle_protocol

Note that the "dtnperf" executable will have either the extension "_vION" or "_vDTN2", if compiled for one specific implementation, or no extension if for both.
Finally, it is possible to install the program in the system directory with the command (with root permissions)

~/dtnperf_3$ make install

DTNperf_3 general description and examples of use.

1. DTNPERF_3 OVERVIEW

DTNperf_3 has three operating modes: client, server and monitor. The client generates and sends bundles, the server receives it and the monitor collects status reports in .csv files. Client, server and monitor correspond to the BP addresses "from", "to" and "reply to".

1.1. DTNperf Client

SYNTAX: dtnperf --client -d <dest_eid> <[-T <s> | -D <num> | -F <filename>]> [-W <size> | -R <rate>] [options]

The most important parameters are explained below. The full list of options can be obtained with the command "dtnperf --client --help".

1.1.1 Tx modes

The client has three mutually exclusive Tx modes to send bundles to the server. In the time-mode (-T), a series of bundles with a dummy payload of desired dimension (-P option) is generated and "sent", i.e. passed to the BP daemon for transmission, until the pre-set transmission time elapses. Data-mode (-D) is the same as time-mode, except that the bundle generation process ends after a given amount of data has been "sent" to BP, and not after a time interval. File-mode (-F) differs from data-mode because a file is transferred, instead of dummy data. A noteworthy feature is the possibility to split the file into multiple bundles of desired dimension.

1.1.2 Congestion control (window- or rate- based)

Independently of the Tx mode, there are two alternative congestion control policies available: window-based (-W) and rate-based (-R). In the former, the "congestion window" W represents the maximum number of bundles in-flight (i.e. sent but not acknowledged). The mechanism is similar to the TCP congestion window, but: 1) W has a fixed dimension; 2) in-flight bundles can be non-consecutive (to cope with non-ordered delivery of BP); 3) there are not retransmissions (acknowledgments are used only to trigger the transmission of new bundles). DTNperf_3 has enhanced this function with respect to previous versions, by replacing "delivered" status reports generated by the BP of the destination node, by ACK bundles specifically created by the DTNperf_3 server, as acknowledgments of bundles sent. This innovation is essential to decouple the "reply to" from the source node, thus allowing the monitor to be launched on a node different from the source.

Although the window-based mechanism is generally useful, in some cases, e.g. to emulate streaming traffic, it would be preferable to generate traffic at constant rate. For this reason, in DTNperf_3 a rate-based congestion control has been added. In response to rate-based traffic the server does not generate any ACKs, like UDP.

1.2. DTNperf Server

The server receives bundles, acknowledges them if sent in window-based mode, and reassembles bundle payloads if a file is transferred.

SYNTAX: dtnperf --server [options]

In this third version, one instance can serve multiple clients in parallel. Moreover, the file transfer is now robust against disordered bundle delivery (incoming payloads are buffered until either the entire file is received or a timeout expires).

1.3. DTNperf Monitor

The monitor receives and collects status reports and some DTNperf control bundles. It can be launched either as an independent process on an external node (external monitor), in which case it can serve many concurrent clients, or as a "child" process of a client (dedicated monitor), in which case it serves only its "parent" client. The syntax to launch an external monitor is the following:

SYNTAX: dtnperf --monitor [options]

The monitor creates a different .csv log file for each DTNperf client session (i.e. one client "launch"). This file contains all status reports generated by all nodes during the session and has been designed to be directly imported into a spreadsheet for further analysis.

1.4. The Abstraction Layer

To facilitate interoperability tests, DTNperf_3 has been designed to be independent of the BP implementation. It relies on the "Abstraction Layer", i.e. a library expressly designed to provide a common interface for DTNperf towards the APIs of different BP implementations.

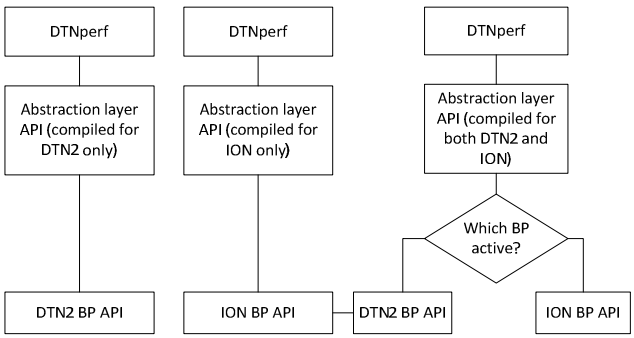

Figure 1: DTNperf compatibility. a) DTN2 only, b) ION only, c) DTN2 and ION with API call selection at run time.

The present version of the AL supports DTN2 and ION, but it could be extended to other implementations. If the AL is compiled either for DTN2 or ION, DTNperf can run only on top of DTN2 or ION (Figure 1a and b). If the AL is compiled for both, DTNperf can run on top of both, as the switch between DTN2 and ION API calls is performed at run time (Figure 1c)

2. DTNPERF_3 USE.

Although derived from authors' experience on space communications, DTNperf_3 has a general scope aiming at embracing most DTN applications. In the examples below, the use of DTNperf_3 in some typical DTN applications is presented, assuming that we have three DTN nodes, vm1, vm2 and vm3 when the dtn scheme is adopted, or nodes 1, 2 and 3 when the ipn scheme is preferred, as source, destination and external monitor. We will focus on client syntax, as the server and the monitor can always be launched with the following commands:

dtnperf - -server - -debug=1

dtnperf - -monitor - -debug=1

2.1. Basic applications

2.1.1 Ping

To ping vm2 from vm1:

dtnperf - -client -d dtn://vm2.dtn -T 15 –W 1 - -debug=1 (if the server is registered with the dtn scheme)

dtnperf - -client -d ipn: 2.2000 -T 15 –W 1 - -debug=1(if the server is registered with the ipn scheme; note that the demux token of the server is always 2000)

With this command vm1 will send bundles of 50 kB (default) to vm2 for 30s (-T15), one by one (-W1 allows just one bundle in flight). The short default lifetime (60s) is useful to force the deletion of undelivered bundles, which could interfere on subsequent experiments. As no external monitor is indicated, the .csv log file will be created on vm1 by a dedicated monitor.

2.1.2 Trace

To trace the route of a bundle:

dtnperf - -client –d dtn://vm2.dtn –D100kB -P100kB –W1 –C - f –r

This command sends a single bundle of 100 kB as the total amount of data (-D100kB) coincides with the bundle payload (-P100kB). The custody option is on (-C) and some optional status reports are requested (forwarded, -f, received, -r). From the .csv file it is straightforward to trace the route of the bundle sent.

2.1.3 File transfer.

To transfer a file segmented into multiple bundles of desired dimension:

dtnperf - -client –d dtn://vm2.dtn –F picture.jpg –P 100kB –W4

Here a file is sent (-F) instead of dummy data. The dimension of the bundle payload (-P100kB) is the dimension of segments into which the file will be split. This feature is useful to match limited contact volumes as an alternative to proactive fragmentation **Errore. L'origine riferimento non è stata trovata.**. Note however that file segmentation is carried out at application layer (by DTNperf), while bundle fragmentation is performed at BP layer (by the BP daemon).

2.2. Performance evaluation in continous and disrupted networks

2.2.1 Goodput (macro-analysis)

Goodput evaluation (i.e. data_ACKed/time), makes sense especially if the DTN network is not partitioned (e.g. in GEO satellite communications, where the challenges are long delay and losses). To this end, it is necessary to send dummy bundles with the window-based congestion control for a reasonable amount of time to reach and maintain a steady state. The following command could be suitable in the case of a hypothetical GEO satellite hop:

dtnperf - -client -d dtn://vm2.dtn -T 30 –P 1MB –W 4 –l 60

With this command vm1 will send bundles to vm2 for 30 s (-T30), i.e. for a much longer interval than the typical GEO RTT (600ms including processing time). To fill the (likely) large bandwidth-delay product and to reduce the impact of bundle overhead, bundles are large (-P1MB) and the congestion window W is greater than one (-W4). The same experiment should be repeated increasing W until the goodput reaches a maximum. Note that goodput evaluation should always be complemented by the analysis of status reports, collected in the example by the internal monitor (default), to control the regularity of the bundle flow and to recognize the reasons of the macro-results achieved.

2.2.2 Status report analysis (micro-analysis).

The evaluation of goodput is useful when links are continuous or only moderately disrupted (e.g. in GEO satellites with mobile terminals). As the chances of disruption increase (e.g. LEO satellites, deep space communications), the study of individual bundles (i.e. micro-analysis) becomes more important than goodput. A possible command in the presence of disruption is:

dtnperf - -client -d dtn://vm2.dtn –m dtn://vm3.dtn –D30MB –P 1 MB –W4 –l 200 (server and monitor registered with dtn scheme)

dtnperf - -client -d ipn:2.2000 –m ipn:3.1000 –D30MB –P 1 MB –W4 –l 200 (server and monitor registered with the ipn scheme; the demux tokens are always 2000 and 1000 respectively)

This command dispatches 30 bundles of 1 MB each, with a limit of 4 bundles in flight. Status reports are collected (possibly in real time by means of dedicated links) by an external monitor (-m option). Note that the lifetime has been increased to 200s (-l200) to cope with disruption. Moreover, Data mode (-D30) is preferable because disruption makes uncertain the actual duration of bundle transfer.

2.2.3 Status report analysis of streaming traffic.

To emulate a streaming source a possible command is:

dtnperf - -client -d dtn://vm2.dtn -T30 –P100kB –R2b

This command generates a stream of 100 kB bundles for 30s, at 2 bundles per second (-R2b). No ACKs are generated by the server, as the congestion control is rate-based.

2.3. **Performance evaluation in partitioned networks: “data mule” communications**

One of the most evident advantages of DTN architecture is the possibility to cope with network partitioning. The extreme case is that of “data mule” communications, where an intermediate node (the “mule”, or “ferry”) is alternatively connected, thanks to its movement, either to the sender or to the destination. In this case the evaluation of goodput is useless, while the micro-analysis is essential. A possible command is:

dtnperf - -client -d dtn://vm2.dtn –m dtn://vm3.dtn –D 10 MB –P1 MB –W10 –l 200 -- debug=1

The external monitor (-m option) is very useful here, especially if connected to other nodes through dedicated links, which can be easily carried out in a lab testbed, since links used to transfer data are only occasionally active. Note that by setting the window to the total number of bundles (10 in the example) these are sent in one burst, which can be preferable in this kind of experiments, in order not to have to wait for ACKs (once the burst of data is sent, the user can interrupt the client by pressing ctrl+c). Alternatively, the user can take advantage of the rate-based congestion control, which does not imply any ACKs (e.g. by setting -R10b instead of –W10).

2.4. **Interoperability tests**

Thanks to the AL library, DTNperf_3 can run on top of either ION or DTN2 BP. Moreover, if the AL is compiled for both, the very same executable can be used. Interoperability tests, however, have also to cope with the different EID schemes preferentially used by DTN2 and ION (“dtn” and “ipn” respectively). To facilitate experiments, DTNperf_3 in ION can register itself also with the dtn scheme, thus allowing full interoperability also with DTN2 nodes unable to use the ipn scheme. In brief, DTNperf_3 can cope with every combination of DTN2 and ION nodes, independently of the EID scheme adopted by these. For example the following command can be launched on a machine running either DTN2 or ION:

dtnperf - -client -d dtn://vm2.dtn –m ipn:3.1000 –D30MB –P 1 MB –W4 –l 200 (server registered with the dtn scheme, monitor with the ipn scheme)

Of course, a prerequisite for DTNperf_3 interoperability is that all DTN2 and ION configuration files are correctly configured for supporting interoperability at bundle layer.