DTN Code for ns-3

The DTN code has been tested with **ns-3.15** [1]. There is no need to re-install ns-3 if ns-3.15 has already been installed. The code may not work with earlier/later ns-3 versions.

A. Install the ns-3.15 allinone package (if needed)

Installing ns-3 is a simple process:

1. Download the allinone package from the following address: http://www.nsnam.org/
2. In some unix machine, in your home directory:
```
tar -xvf ns-allinone-3.15.tar.bz2
cd ns-allinone-3.15
./build.py --enable-examples --enable-tests
cd ns-3.15
./waf clean
./waf -d optimized --enable-examples --enable-tests configure
./waf clean
./waf -d debug --enable-examples --enable-tests configure
./waf
./test.py -c core
```
3. For further reference:
http://www.nsnam.org/docs/release/3.15/tutorial/singlehtml/index.html

B. Install and test the DTN code

Adding the DTN code on top of basic ns-3 is quite straightforward:

1. In your home directory, create new directory for the DTN code and extract the files there:
```
mkdir ns-allinone-3.15/ns-3.15/examples/DTN_SF_UDP
mv ns3dtn.tar.gz ns-allinone-3.15/ns-3.15/examples/DTN_SF_UDP
cd ns-allinone-3.15/ns-3.15/examples/DTN_SF_UDP
tar -xvf ns3dtn.tar.gz
```
2. Move the modified regular-wifi-mac.h (modified for DTN code), dcf-manager.cc (bug fix) and mac-low.cc (bug fix) files to ns-allinone-3.15/ns-3.15/src/wifi/model directory:
```
mv regular-wifi-mac.h ../../src/wifi/model
mv dcf-manager.cc ../../src/wifi/model
mv mac-low.cc ../../src/wifi/model
```
3. Subdirectories Run1-Run6 as well as files dtn.cc, mypacket.cc, mypacket.h, simulate.sh, waf, and wscript stay in the current directory. Move the trace file, sf_downtown_nodes116_time3600.tcl, to ns-allinone-3.15/ns-3.15/examples directory:
```
mv sf_downtown_nodes116_time3600.tcl ..
```
4. Run the example script (San Francisco cab mobility traces, 116 nodes, 3600 seconds, 5700 m times 6600 m area) in the current directory:
```
nohup ./simulate.sh &
```

C. Try out the simulation campaign

Download the other gzipped archive and extract it in ns-allinone-3.15/ns-3.15/examples directory (after you have installed the DTN code):

1. Extract the files:
```
tar -xvf ns3dtn_campaign.tar.gz
```
2. Modify the simulate.sh files (explicit paths to working directory) in all eight directories and then launch the simulations:
```
cd DTN_SF_UDP
```

```
emacs simulate.sh
nohup ./simulate.sh &

cd ../DTN_SF_SW_UDP
emacs simulate.sh
nohup ./simulate.sh &

cd ../DTN_SF_SW_CC_UDP
emacs simulate.sh
nohup ./simulate.sh &

cd ../DTN_SF_CC_UDP
emacs simulate.sh
nohup ./simulate.sh &

cd ../DTN_SF_CC_0_7_UDP
emacs simulate.sh
nohup ./simulate.sh &

cd ../DTN_SF_CC_0_9_UDP
emacs simulate.sh
nohup ./simulate.sh &

cd ../DTN_SF_CC_Ad_UDP
emacs simulate.sh
nohup ./simulate.sh &

cd ../DTN_SF
emacs simulate.sh
nohup ./simulate.sh &
```

3. When the simulations are over (in a few days, depending on your hardware), start Matlab in ns-allinone-3.15/ns-3.15/examples directory and plot the results with the script that is included in the package:
   ```
   matlab -nodesktop
   >> dtn_cc_trace_SF_comp_tcp_udp
   ```

This simulation campaign tests congestion control with both epidemic routing and spray-and-wait routing in realistic environment (San Francisco cab mobility traces). See the section 4.3 of the paper for details.

D. How to use/modify the DTN parameters

Default values of the DTN variables

- Hello message interval: hard coded to 100 ms.
- Bundle lifetime: hard coded to 750 s.
- Antipacket lifetime: hard coded to min(750, 1000 - bundle travel time) s.
- Bundle retransmission timeout: hard coded to 1000 s.
- Number of bundle retransmissions: 3.
- Use of antipackets: yes (hard coded).
- Bundle routing protocol (antipackets are always sent using epidemic); 0 = epidemic, 1 = binary spray and wait:
  ```
  rp (0)
  ```
- Maximum number of bundle copies in binary spray and wait: hard coded to 16.
- Bundle drop strategy: hard coded to drop tail.

- Bundle buffer size [bytes]:
  `b_s (1000000)`
- Congestion control: off (0), static (1) or adaptive (2):
  `cc (0)`
- Congestion control threshold:
  `t_c (0.8)`

Counters that can be used

- Antipacket queue size (maximum size: 1000 packets, bundle buffer size does not limit this):
  `m_antipacket_queue->GetNPackets ()`
- Bundle queue size :
  `m_queue->GetNBytes ()`
- Number of neighbor nodes:
  `neighbors (0)`

Mobility files (ns-2 format that is converted to ns-3 format)
`sf_downtown_nodes116_time3600.tcl, centre_MovementNs2Report.txt`

E. Main functions in dtn.cc

- `SendHello`: Broadcast a Hello message that contains the identifiers of those bundles that are stored in this node as well as advertised free buffer capacity of the node. The size of the Hello message depends on the number of bundles stored in the node. Hello messages have priority over all other messages; this is implemented with quality of service on MAC level.
- `ReceiveHello`: Receive Hello messages and update neighbor information, e.g., what bundles they currently have.
- `CheckBuffers`: Periodically check what bundles and receipts in the storage can be forwarded.
  - Remove all expired bundles from the storage.
  - Check if the MAC queue occupancy is below a given threshold (currently hard coded: two packets). If this is the case, we can proceed and select the next packet to be transmitted. Antipackets have priority over regular bundles.
  - Congestion control can be applied here: if the bundle storage occupancy in the neighbor node is above the congestion control threshold, we do not forward bundles to this node.
  - Both UDP and TCP (code included in the ns3dtn_campaign.tar.gz package) can be used for bundle transmission. In the UDP case, acknowledgements (one ack per packet) and retransmissions (retransmission timeout: 1.0 seconds) are applied. With UDP-based transmission, we check before sending each packet if the neighbor is still there. If the neighbor is not within transmission range, we shall continue sending when we are close enough.
- `SendBundle`: Send a bundle with a given lifetime to a given neighbor.
- `ReceiveBundle`: Receive bundles and process them accordingly.
  - First check if all packets of a bundle have been received; if yes, then forward/receive/delete the bundle. Tail dropping is applied if bundle storage is full.
- `SendAP`: Send an antipacket/return receipt to a given neighbor with a lifetime depending on bundle delay.

- `ReceiveAP`: Receive antipackets/return receipts and process them accordingly.
    - Delete the corresponding bundle from storage and add its identifier to the database (so that possible copies can be deleted when they arrive).

F. References

[1] ns-3 web site," Sep. 2012. http://www.nsnam.org/

[2] J. Lakkakorpi, M.J. Pitkänen, and J. Ott, "Using Buffer Space Advertisements to Avoid Congestion in Mobile Opportunistic DTNs," in Proc. WWIC 2011, Barcelona, Spain, Jun. 2011.