Generic Opportunistic Routing Framework

draft-lindgren-dtnrg-gorf-00
draft-lindgren-dtnrg-gorf-epidemic-00
draft-lindgren-dtnrg-gorf-prophet-00

Routing in DTNs

- Most popular DTN research topic
  - 100+ papers
- However, only two protocols specified as nice implementable IDs
  - PRoPHETv2 (now RFC6693)
  - dLife
- Makes it hard to implement a protocol
- Difficult to evaluate different algorithms

Common framework

- A lot of the routing algorithms have a lot of similarities
  - Exchange some routing information and a list of bundles with nodes you meet
  - Use some utility function or other mechanism to decide which bundles to exchange
- A lot of work was done with PRoPHET to go from research paper to RFC
- Not everyone should have to repeat this process

Generic Opportunistic Routing Framework

- Cut out all the surrounding protocol mechanisms that are the same for most protocols and put this in a common framework.
- Each new routing algorithm only has to define a routing algorithm module specification
- Much shorter document that is easier to write and to read/understand
- Implementation becomes easier

Benefits

- Quick and easy to create an implementable routing protocol spec
- Fast prototyping and testing
  - Once the framework is implemented, it is simple to create a new module for a new routing algorithm
    - Just a new class/module/library
- Instead of everyone just simulating on their own, easier to test many protocols, and (more importantly) test it in real systems.

Routing algorithm module

- Each new routing algorithm module needs to define the following:
- Internal state:
  - Routing metric format
    - Hop counts, delivery predictability, geo info, etc
    - (compact format to describe (most possible) metrics included)
      - 8/16/32/64 bit integers, SDNVs, Strings, bit sequences, lists
  - Node characteristics format
    - Any info about that nodes want to spread about themselves in addition to routing tables
    - (location, battery power, remaining buffer space, etc)
  - Any other internal state needed by the algorithm

Routing algorithm module (2)

- Definitions for 14 functions for…
- …keeping track of routing metrics and node characteristics:
  - `getMetricFormat()`
  - `getNodeCharacteristic()`
  - `getNodeCharFormat()`
- …meeting new nodes and exchanging routing info:
  - `encounteredNode()`
  - `getRoutingState()`
  - `updateRoutingState()`
- …determining which bundles to exchange:
  - `generateOffer()`
  - `generateResponse()`
  - `bundleSent()`
- …maintain and use long-lived connections:
  - `newBundleArrived()`
  - `informationExchangeTimer()`
  - `nodeDisconnected()`
- …buffer management:
  - `dropAdvice()`
  - `ackReceived()`

Example modules:

- Epidemic Routing:
  - draft-lindgren-dtnrg-gorf-epidemic-00
  - 8 pages
  - (mostly ID stuff, 3 pages of ”real content”)
- PRoPHET(v2)
  - draft-lindgren-dtnrg-gorf-prophet-00
  - To appear soon….
  - Y pages
  - As compared to 113 page of RFC6693…
    - (of course, most of those pages are now in the framework document, but nice to not have to repeat that for every new routing algorithm)
- dLife
  - Previously discussed with Paulo Mendes to make dLife run as part of this framework
  - (Not confirmed since the release of the draft)

Future potential enhancements

- Different destination/node/message formats
- Should we always use dictionaries and SDNVs?
- Go through the different protocol details and re-consider some…
- Change and add stuff based on feedback
  - In particular from people saying “I need this functionality to implement algorithm XYZ”
- Contributions from others to the main document
  - Preliminary discussions with Paulo Mendes