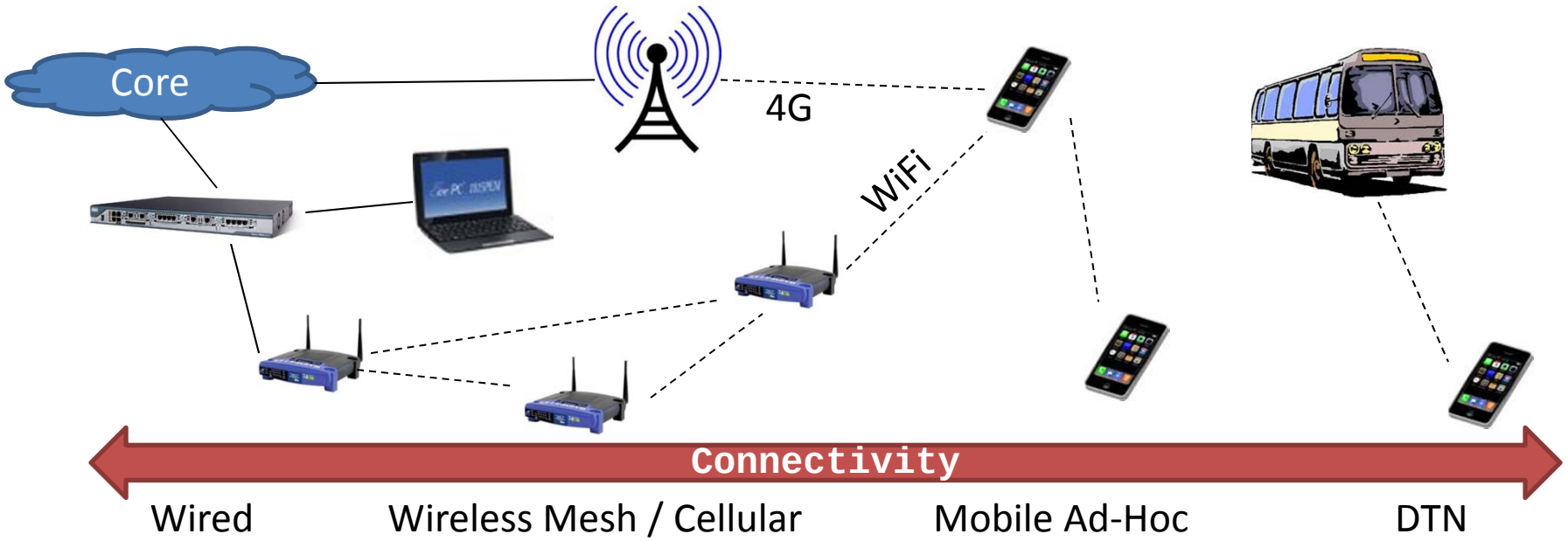
MobilityFirst

GSTAR: Generalized Storage-Aware Routing

Samuel Nelson

MobilityFirst Design Goals

- Design a future internet architecture that supports:
  - *Host and network mobility*
  - *Diverse communication devices/entities/paradigms*
  - *Strong security and privacy*
  - *Large data units, as opposed to flows (e.g., no state)*



Challenges of Mobility

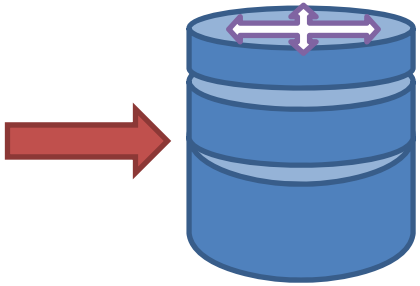
- Local-scale mobility challenges
  - Large variance in node mobility
  - Link quality can quickly fluctuate
  - Congestion can become concentrated
  - Nodes can become disconnected from the network
  - The network itself can become partitioned

The *network layer* is in the best position to directly deal with these topological problems

Let's give routers the resources they need to handle these challenges!

In-network Storage

Take advantage of cheap storage in the network (*storage-aware routing*)



~100MB, data in transit

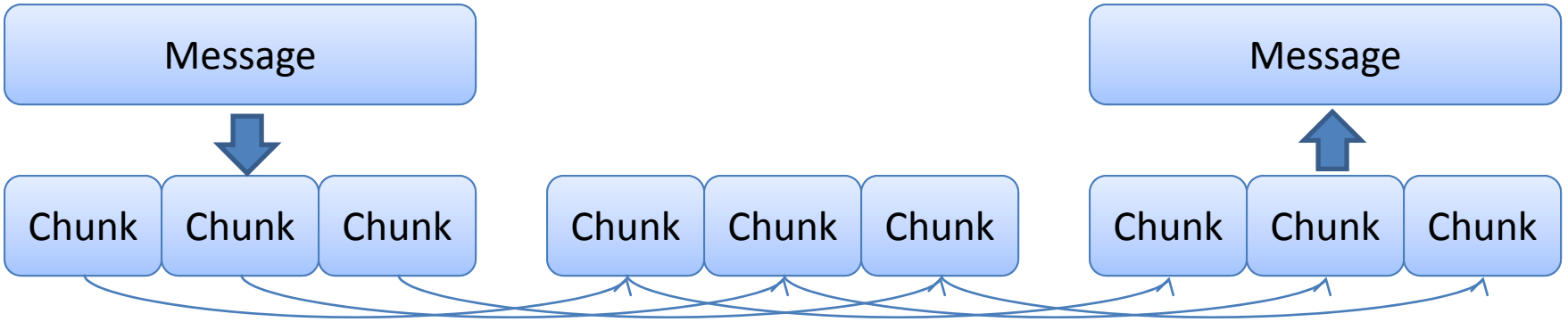
~10GB, in-network storage

~1TB, content caching

Mobility Challenges	Storage-aware Solutions
Link quality fluctuation	Proactively hold messages
Node disconnection	Proactively push data to likely reattachment points
Network partition	Allow nodes to ferry messages across partitions
Congestion	Storage-aware path selection; proactively hold messages

Hop-by-hop Transfer

- Storage allows for *hop-by-hop* transfer of large data chunks
  - Similar to the DTN bundle, a chunk is an autonomous unit of routable data
  - Breaks away from end-to-end streaming protocols and goes back to pure packet-switching mechanism
- Advantages:
  - Robust to mobility, opportunistic delivery, decreased overhead, less “chatter”



Intra-domain Routing

- R3 (led by Arun at UMass)
  - Bridges wireless domains, from mesh to DTN
- GSTAR
  - Fundamentally *storage-aware* link-state, with flexible path selection and transmission decisions due to storage availability
  - Augmented with DTN capabilities at the GUID level, with the ability for nodes to carry messages, again due to storage availability

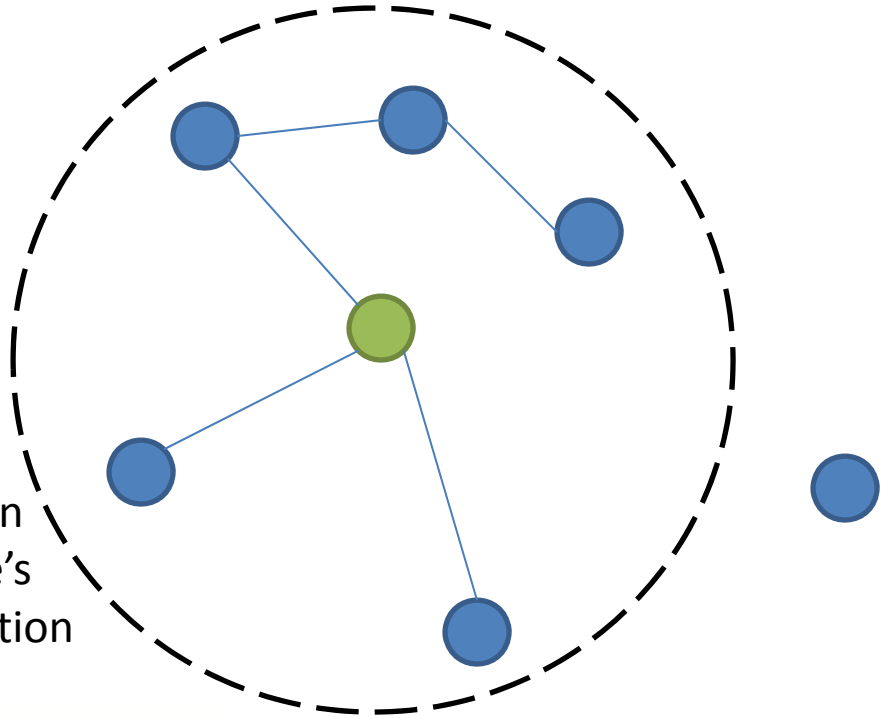
High-Level Overview

Give all nodes *within* your partition fine-grained, time-sensitive topology information

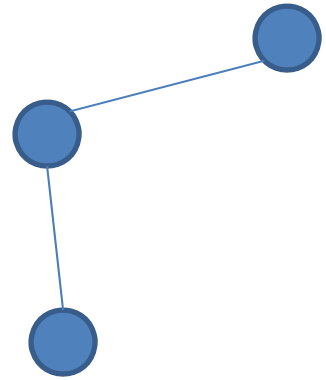
Works on *local addresses (LAs)*

Give all nodes in the network course-grained, time-insensitive connectivity information

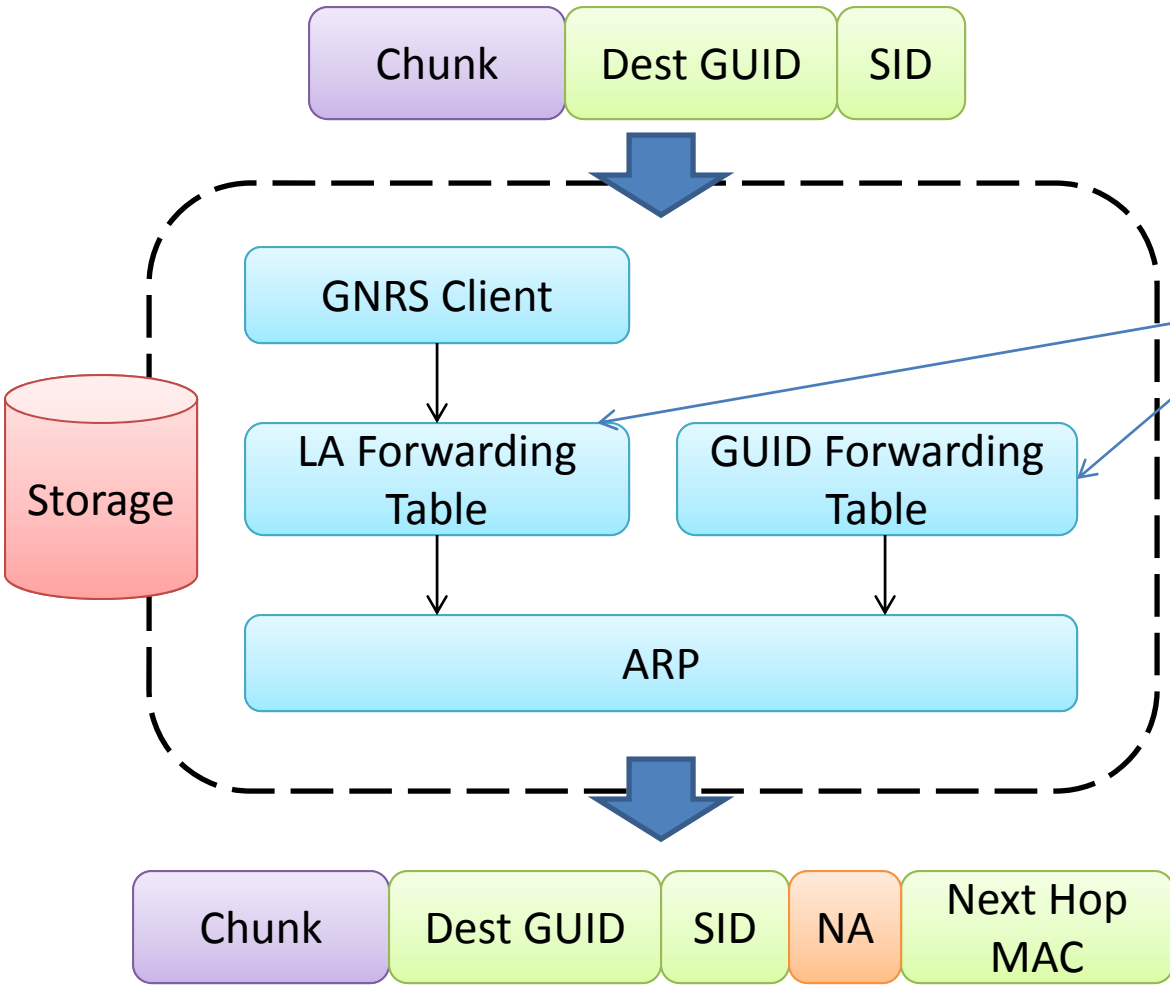
Works on *GUIDs*



Green node's partition



Routing Architecture



Populated by proactively disseminating two types of control information

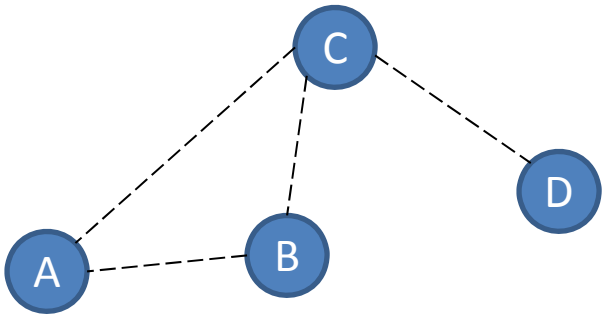
Storage-aware path selection generates these tables

Control Messaging

- All routers participate in disseminating two types of topology information: (1) specific, time-sensitive and (2) general, course-grained

Time sensitive metric

Flood short and long term link quality estimates for neighbors (reaching intra-partition nodes)

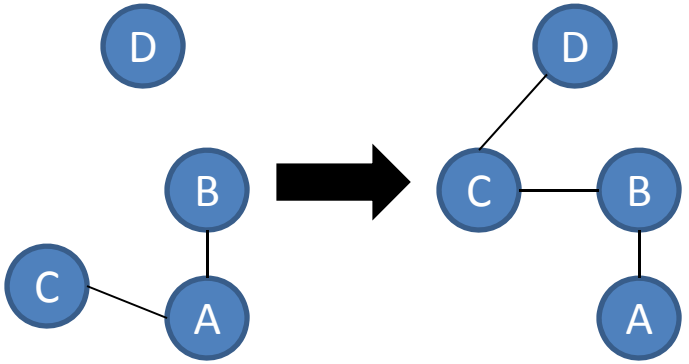


D sees:

IP_A <-> IP_C (fair/fair)
IP_A <-> IP_B (good/fair)
IP_B <-> IP_C (fair/bad)
IP_B <-> IP_C (good/good)
IP_C<-> IP_D (fair/fair)

Time insensitive metric

Epidemically disseminate GUID-based contact probability information (reaches all nodes) *



D sees:

A <-> C (1/2)
A <-> B (1)
B <-> C (1/2)
C <-> D (1/2)

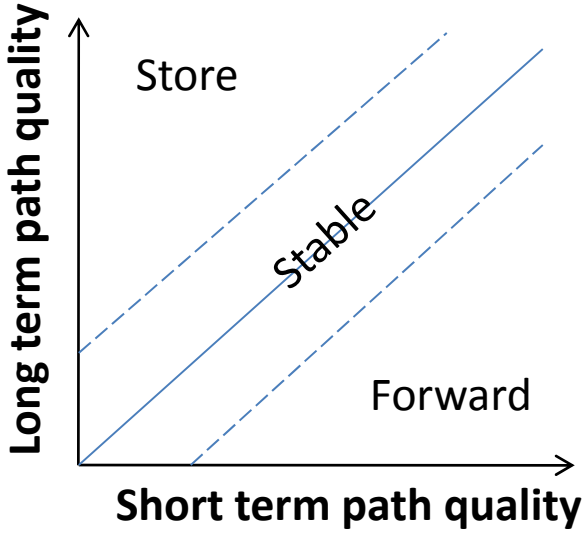
* Similar to *R Ramanathan, et al. Prioritized Epidemic Routing for Oppurtunistic Networks*

Where is the Destination?

1. GNRS client makes a GUID-based request and determines if
   - The destination definitely within my network (so, attach an LA)
   - The destination is definitely not within my network (so, attach an NA) Use gateway
   - I am not sure where the destination is (attach nothing) Use GUID Use gateway
2. If an LA is returned, consult the *LA forwarding table* to see if
   - There is a valid entry Use LA
   - There is not a valid entry Use GUID

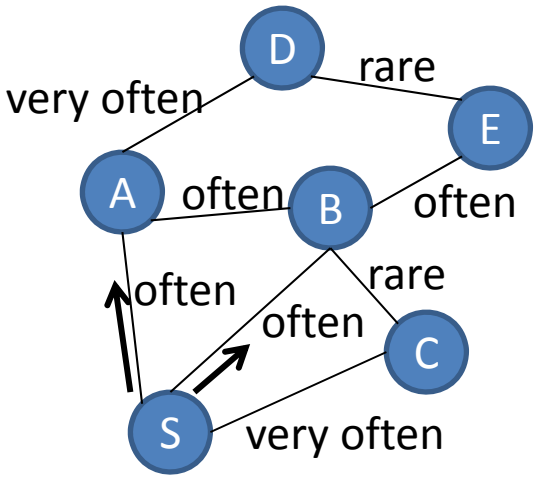
LA-based Data Forwarding

- If using the *local address*
  - Compute a path-based short and long term quality
  - Store if the short term path quality is *abnormally bad*
    - SPQ > 1.1 x LPQ
- Proactively storing helps alleviate both storage and "airway" congestion



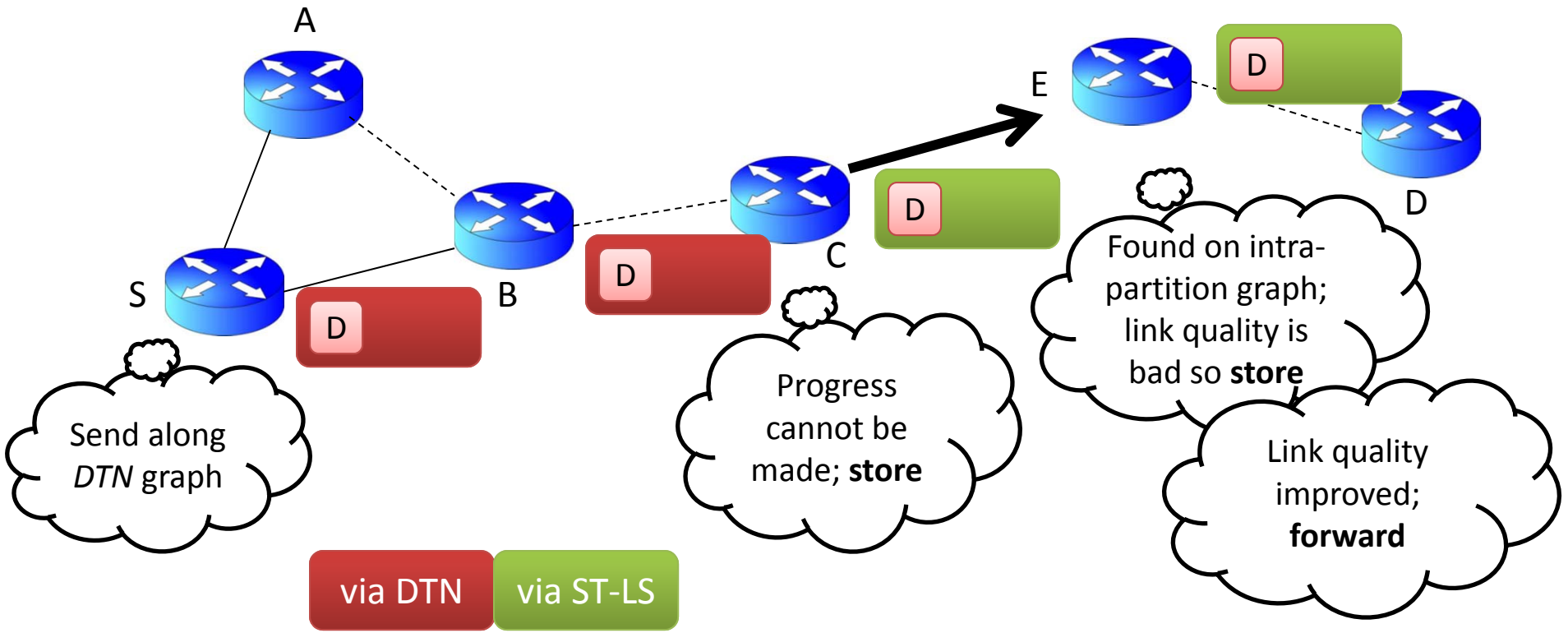
GUID-based Data Forwarding

- If using GUIDs
  - Send data to the node that makes progress along the *DTN graph*
  - Weights along this graph represent connection availabilities between GUIDs
  - Replication is possible here, and we are currently exploring this option



Pure GUID-based Routing

- It is also possible to do all routing on the flat GUID space, without the use of a GNRS



Storage-Aware Path Selection

- Goal:
  - Choose lowest delay paths, taking into account storage possibilities

Delays at each router
1. Store block until ready to send
2. Gain access to the channel
3. Transmit the block

$$delay_{1,2,\ldots,k} = \sum_{i=1}^{k} \left(P_i \cdot HOLD_i + BACK_i + CHAN_i + T_i \right)$$

Prob. of store decision & expected time in storage

Waiting for buffer to clear

Gain access to channel

Time to send chunk

Securing Control Messages

- Attack:
  - Modify fields in control messages not belong to you
- GSTAR is fundamentally link-state, and hence no control message requires modification or aggregation (unlike distance or path vector)
  - No mutable fields in control messages
- Therefore, a control message can easily be *signed* by the entity sourcing it, using the $GUID^{-1}$ key

Storage-Based Attacks

- If storage availability is used as a metric in path selection, then:
  - Malicious nodes can announce *infinite storage* and redirect all traffic through them
  - Malicious nodes can fill buffers on parallel paths and redirect all traffic through them
- Possible solution:
  - Limit the amount of influence a single node has on the path storage metric
    - Average is bad (1 node changes everything)
    - Median is better (need at least n/2 nodes to arbitrarily change
  - Messages going into storage must be signed so they can be kept track of