Message Drop and Scheduling in DTNs: Theory and Practice

Chadi BARAKAT, Ph.D.

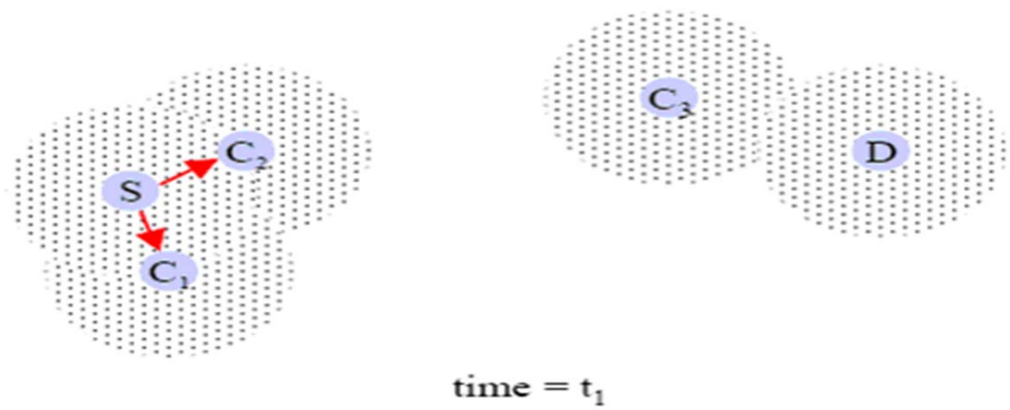
INRIA Sophia Antipolis, France
Planète research group

Joint work with Amir Krifa, INRIA
Thrasyvoulos Spyropoulos, EURECOM

Email: Chadi.Barakat@sophia.inria.fr
WEB: http://www.inria.fr/planete/chadi

The DTN principle

- Rely on node mobility to route messages through disconnected networks
  - A node can be a human carrying a laptop or PDA, a bus, a car, a satellite, etc
  - Nodes carry messages of each other while moving
- At the opposite of existing networks, no end-to-end path is required during the communication
  - Hop-by-Hop networking compared to existing end-to-end paradigm
  - Message vs packet
  - Message replication and forwarding between DTN nodes



INRIA SOPHIA ANTIPOLIS

Many applications

- Space communication
  - Satellites move. Not always in sight of each other.
- Sensor network (e.g. zebranet)
  - Sensors that move and that need to send data to sinks.
- VANET: Vehicular networks.
  - Cars that communicate.
- Pocket switched networks:
  - PDAs and laptops that communicate.
- Internet to nomadic commuities
  - Caravans, buses, boats, etc.
- etc

An important block: DTN routing

- The block that decides whether to replicate a msg or not
- An important tradeoff:
  - The more the copies the more the chance
  - But the more the load on the network
- Different options:
  - Global optimal: when routes are known a priori (the rural case)
  - Epidemic: give a copy to everyone
  - Utility-based: copy to those that have more chance to reach soon the destination (or that are good relays)
  - Spray-and-wait: Limit the number of copies in the network

Scheduling and Drop in DTNs
Another important component

- An intelligent routing limits the number of copies, but is not the final solution …
- Nodes' buffers can still overflow due to many messages
  - Which message to drop in case of congestion ?
    - Last In ? First In ? Oldest ? Youngest ? Other ?
- Contact times can be shorter than what is needed to exchange messages between nodes
  - In there is a way to forward the most useful messages first ?
- Two important and still open problems

Outline of the talk

- A analytical framework
- Optimal solution that requires global knowledge
- Distributed version that works in practice
- Validation by simulations and real traces
- Conclusions

Methodology

- Consider point-to-point communications (can be generalized)
- Suppose first global knowledge
- Take a global routing metric as the delay or delivery rate
- Find what is the best policy to drop and schedule
  - Drop locally (or schedule first) the message that leads to the best marginal gain in the considered global metric
  - Model this gain as a <u>per-message utility</u>
  - Our optimization is then local (or greedy)
- Try to estimate the global knowledge using global information BUT on old messages …

Some notations

$K(t)$	Number of distinct messages in the network at time t,
TTL_i	Initial Time To Live for message i,
L	Number of nodes in the network,
R_i	Remaining Time To Live for message i,
$T_i = TTL_i - R_i$	Elapsed Time for message i. It measures the time since this message was generated by its source,
$n_i(T_i)$	Number of copies of message i in the network after elapsed time Ti,
$m_i(T_i)$	Number of nodes (excluding source) that have *seen* message i since its creation until elapsed time Ti.
λ	Meeting *rate between two nodes* *= 1 /* average meeting time

Case of delivery rate

- Suppose each message has limited lifetime (TTL)
  - Our framework is general enough to allow different TTLs
- Suppose global knowledge on the number of copies per message (estimators to be presented next)
- Global delivery rate at the time of congestion is then:

$$DR = \sum_{i=1}^{K(t)} P_i = \sum_{i=1}^{K(t)} \left(\left(1 - \frac{m_i(T_i)}{L-1}\right) * \left(1 - \exp\left(-\lambda n_i(T_i) R_i\right)\right) + \frac{m_i(T_i)}{L-1} \right)$$

Assumption: meeting times have an exponential tail
(known to be a reasonable assumption after some mixing time)

Case of delivery rate

We differentiate:

$$\Delta(DR)=\sum_{i=1}^{k(t)}\frac{\partial P_i}{\partial n_i(T_i)}*\Delta(n_i(T_i))=\sum_{i=1}^{K(t)}\left(1-\frac{m_i(T_i)}{L-1}\right)\lambda R_i\exp\left(-\lambda n_i(T_i)R_i\right)*\Delta(n_i(T_i))$$

The best message to drop is the one having the minimum partial derivative:

$$\left(1-\frac{m_i(T_i)}{L-1}\right)\lambda R_i\exp\left(-\lambda n_i(T_i)R_i\right)$$

And the message to schedule first is the one maximizing it

This is a function of global information on message i. We call it per-message utility relative to delivery rate

We call the resulting policy GBSD (Global knowledge based scheduling and drop)

Case of delivery delay

In the same way, one can find the per-message utility relative to the global delivery delay:

$$\frac{1}{n_i^2(T_i)\lambda} * \left(1 - \frac{m_i(T_i)}{L-1}\right)$$

To minimize the global delivery delay:

- Drop the message having the smallest utility
- Schedule first the message having the largest utility

For more details:

Amir Krifa, Chadi Barakat, Thrasyvoulos Spyropoulos, "Optimal Buffer Management Policies for Delay Tolerant Networks", in proceedings of the SECON conference, San Francisco, June 2008. A detailed version to appear in IEEE TMC.

Some observations

- The optimal decision is function of the number of copies of a message and its remaining lifetime $\left(1-\frac{m_i(T_i)}{L-1}\right)\lambda R_i \exp\left(-\lambda n_i(T_i) R_i\right)$
  - Not equivalent to any simple policy:
    Drop Tail, Drop Front, Drop Youngest and Drop Oldest
- A node needs to know the global information on the messages present in its buffer
- Note that our policy does not make any assumption on the underlying routing protocol

Heterogeneous mobility

- Nodes might have different speeds or different interactions with each other
- They might meet at different rates λ
- The extension is straightforward

For Delivery Rate, per-message utility becomes

$$\left(1-\frac{m_i(T_i)}{L-1}\right).\ln(\Gamma(R_i)).(\Gamma(R_i))^{n_i}$$

where $\Gamma(R_i)$ is the Laplace Transform of λ at point R_i

Optimality

- We are trying to solve the following optimization problem

  Best copy allocation that maximize global metric

  While satisfying storage resource constraint

- We don't solve this problem at once
- But iteratively, with one step towards the optimum upon each contact between nodes
- We keep tracking any change in the optimal allocation
  - Because new messages arrive
  - And others are delivered

Distributed version: How to calculate n and m ?

- n = number of copies of a message
  m = number of nodes that have seen the message
- Flooding (like in RAPID by UMASS) does not work because it takes long time to converge
  - The information is stale by the time it reaches everyone
- Our solution:
  - Still flood information on messages
  - Estimate n and m from what happened to old messages at the same elapsed time
  - Assuming of course a stationarity of the order of flooding time

Distributed version (ctd)

- Requirements:
  - Estimators are to be plugged in the utility expressions
  - We want the global network performance to be preserved
    - Same average delivery delay and same average delivery rate

- Suppose m and n follow two random variables M and N

  Take for example the delivery rate. Estimators should satisfy:

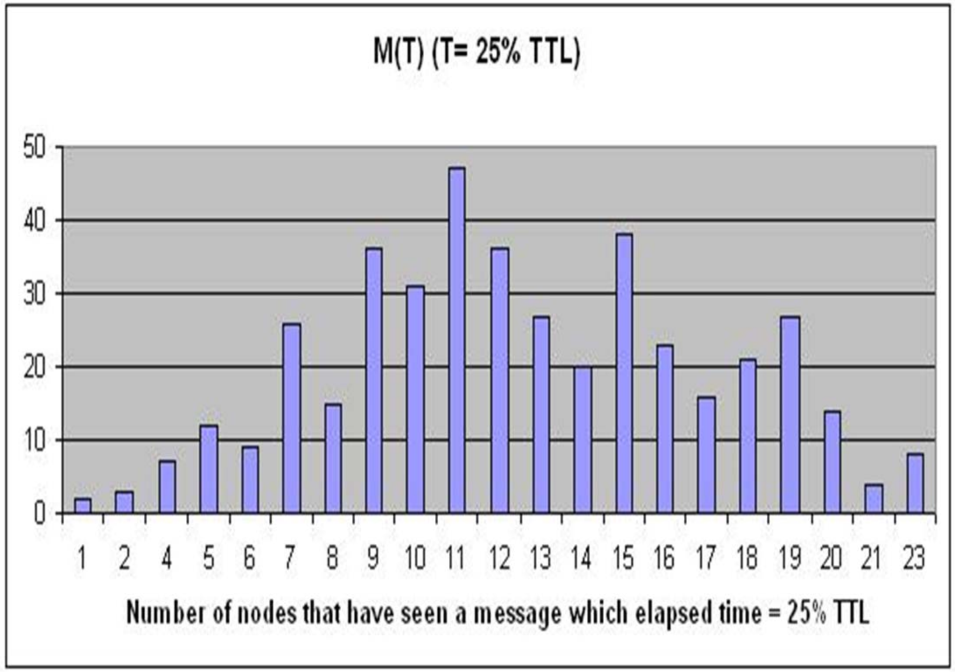
mean delivery rate = estimated delivery rate

$$E\left[\left(1-\frac{M(T)}{L-1}\right)*\left(1-\exp\left(-\lambda N(T)R_i\right)\right)+\frac{M(T)}{L-1}\right]=\left(1-\frac{\hat{m}(T)}{L-1}\right)*\left(1-\exp\left(-\lambda\hat{n}(T)R_i\right)\right)+\frac{\hat{m}(T)}{L-1}$$

Distributed version (ctd)

- We set the estimator of m to its expectation (justified by a Gaussion distribution) $\hat{m}(T) = \overline{m}(T) = E\left[M\left(T\right)\right]$
  - Another value can be used
- We solve the previous equation to get the estimator of n:

$$\hat{n}(T) = -\frac{1}{\lambda R_i} * Ln\left(\frac{E\left[\left(1-\frac{M(T)}{N-1}\right) * \exp\left(-\lambda R_i N(T)\right)\right]}{\left(1-\frac{\overline{m}(T)}{L-1}\right)}\right)$$



- Then we plug in the per-message utility expression

Distributed version: Message utility expressions

For the delivery rate:

$$\lambda R_i E\left[\left(1-\frac{M(T)}{L-1}\right)*\exp\left(-\lambda R_i N(T)\right)\right]$$

For the delivery delay:

$$\frac{E\left[\frac{L-1-M(T)}{N(T)}\right]^2}{\lambda*\left(L-1\right)*\left(L-1-\overline{m}(T)\right)}$$

History Based SD (HBSD)

Expectation calculated by summing over old messages

Experimental results: Setup

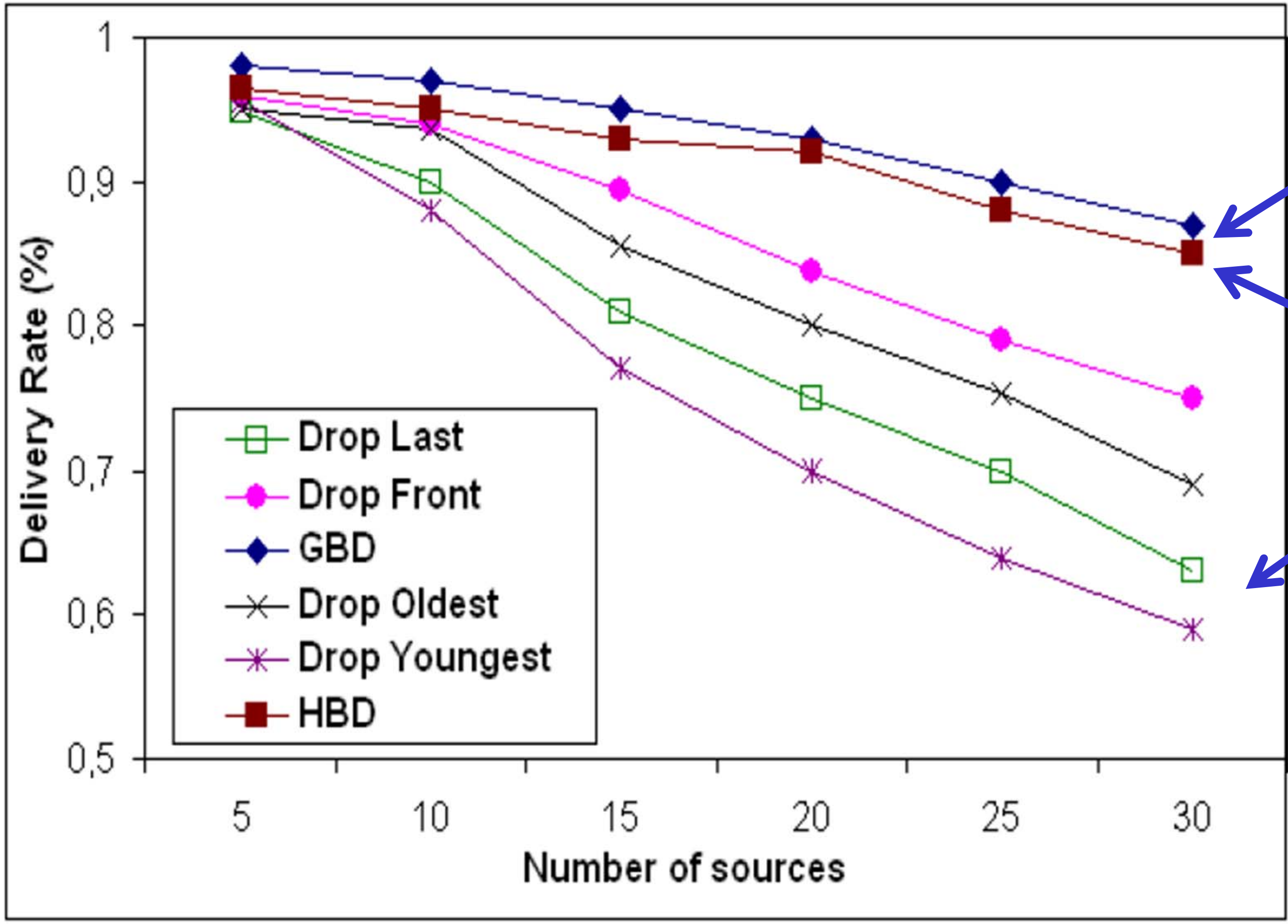
Mobility model	Random Waypoint	Traces du projet ZebraNet	Traces du projet Cabspotting
Simulation duration (s):	5000	5000	36000
Simulated Surface (m^2):	1000*1000	1500*1500	-
Number of nodes:	30	40	40
Average speed (Km/h) :	6	-	-
TTL (s) :	650	650	7200
Intervalle CBR (s) :	200	200	2100

DTN architecture added to the ns-2 simulator

MAC = 802.11b, range=100m, CBR sources, random sources and destinations

Delivery Rate

Messages of 1Kbytes each (Random Way Point)

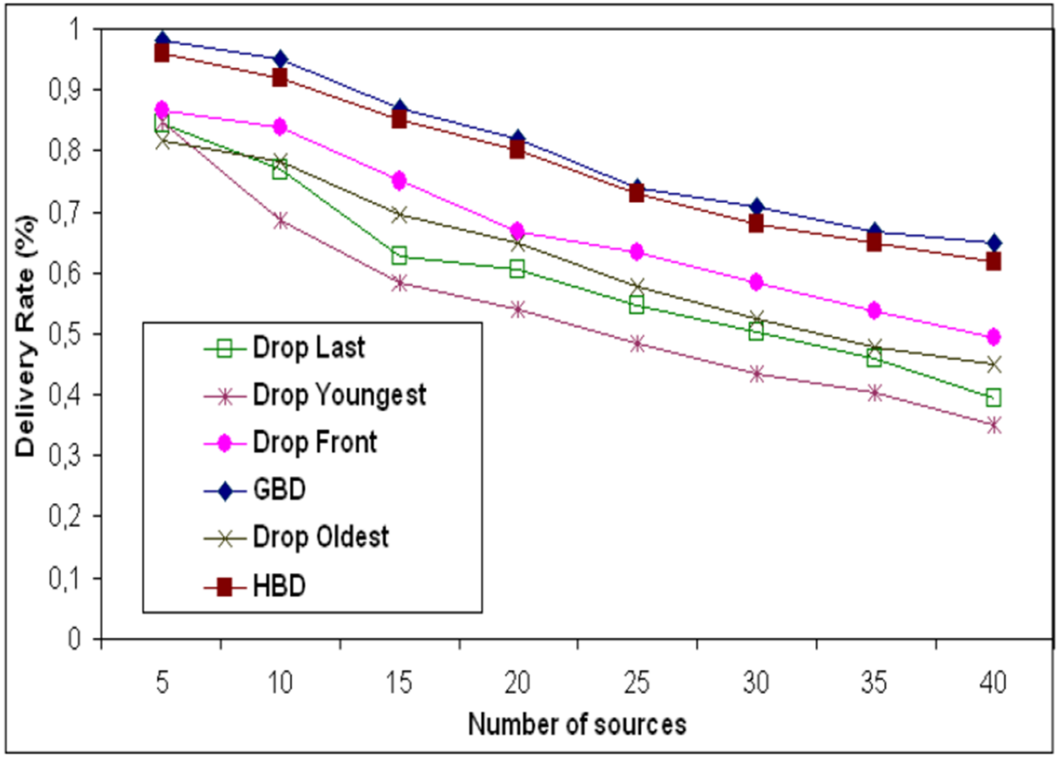


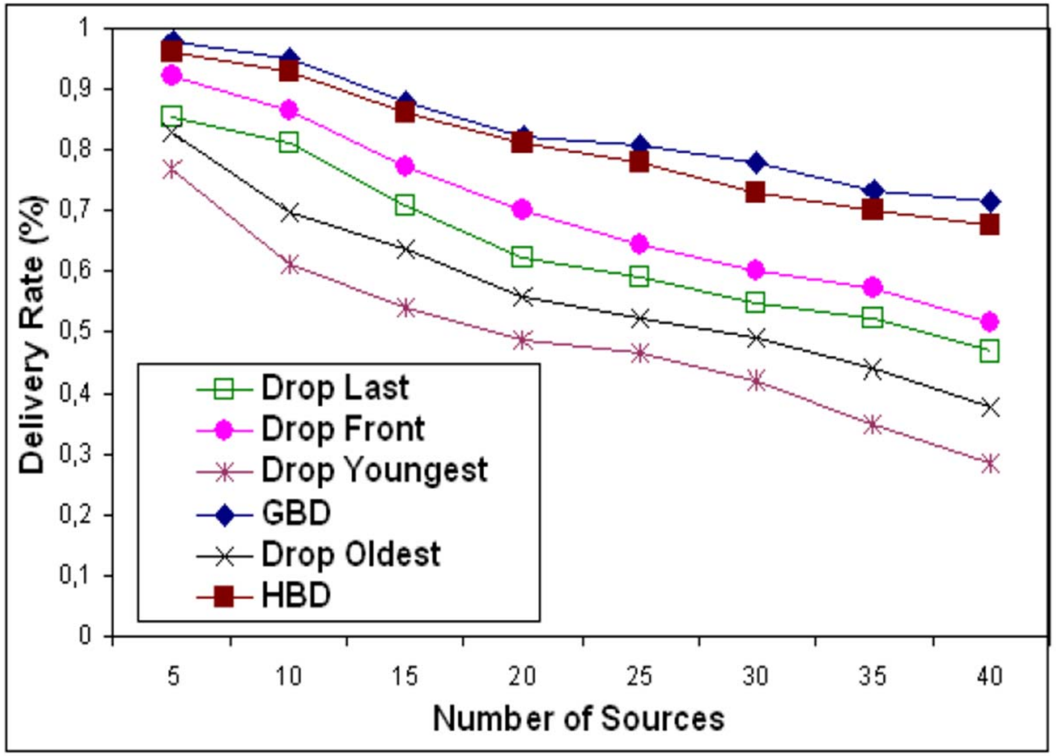
Very close

Almost 50% gain over DropTail

Delivery rate

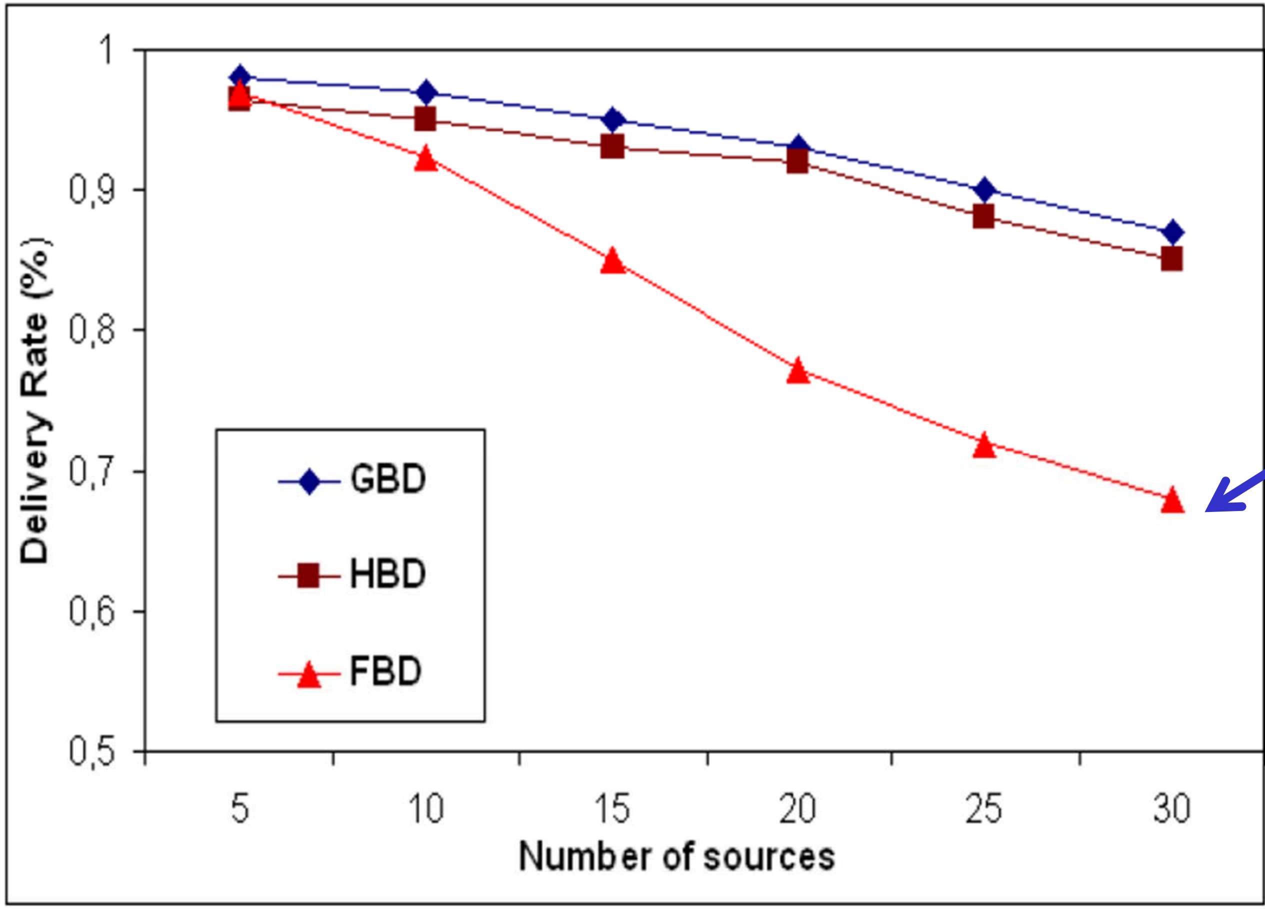
Messages of 1Kbytes each (Real Traces)





Flooding vs. History

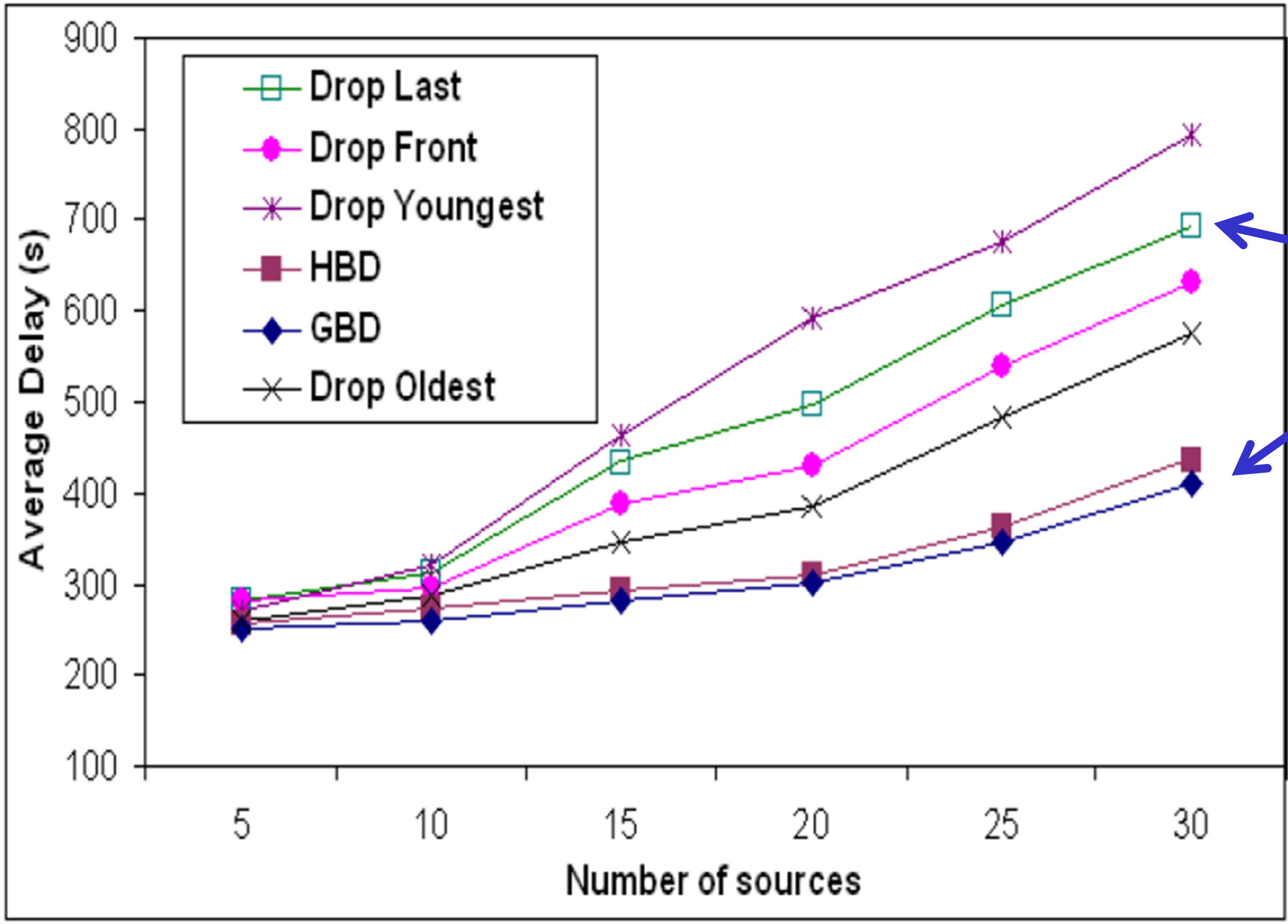
Messages of 1Kbytes each (Random Way Point)



RAPID

Delivery Delay

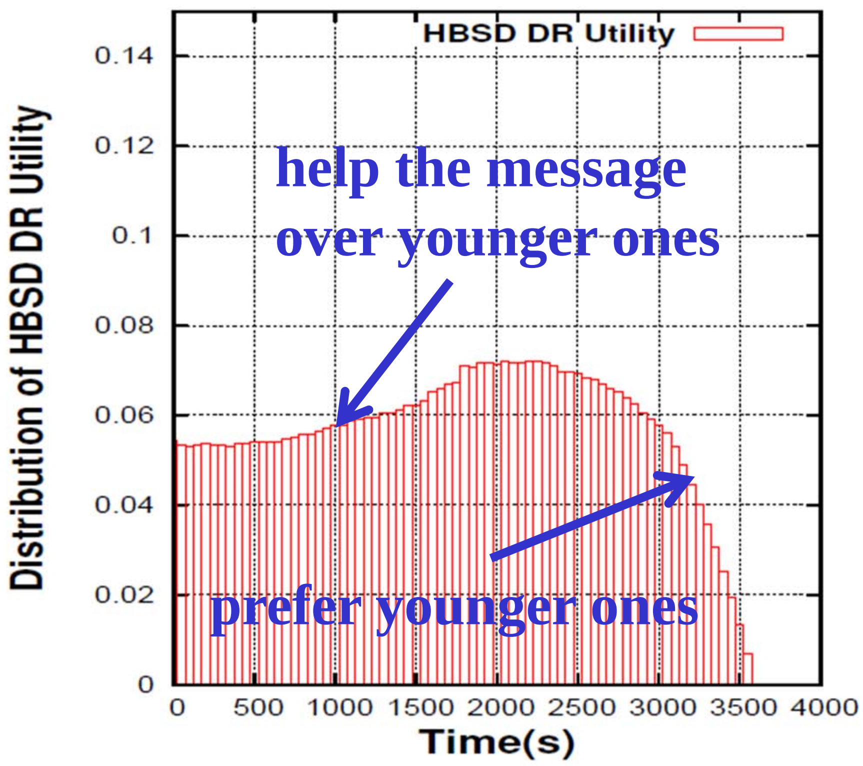
Messages of 1Kbytes each (Random Way Point)

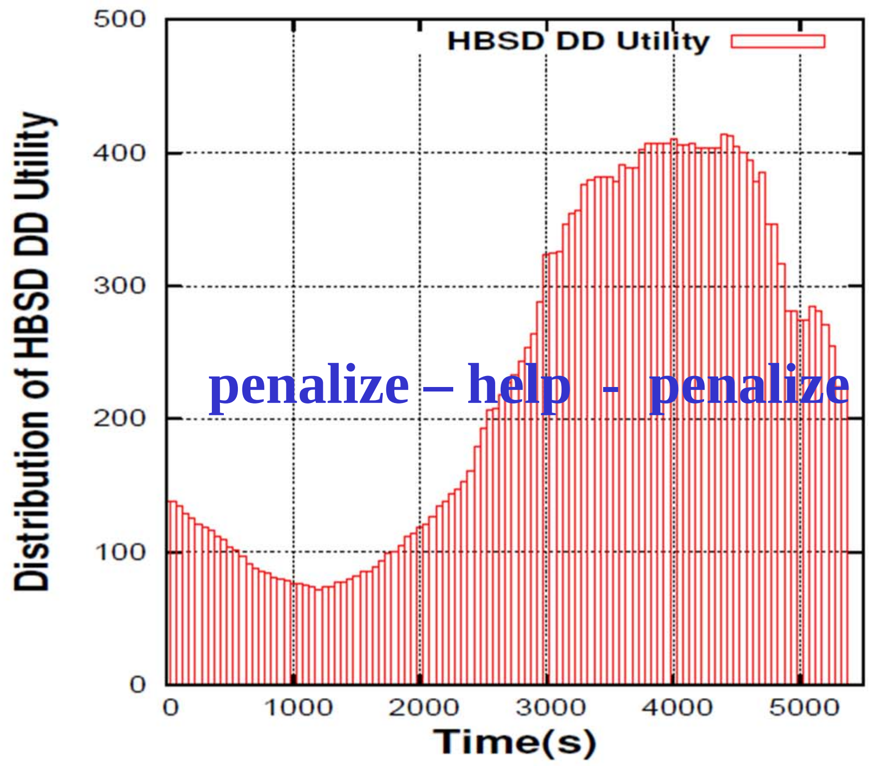


Again, almost 50% gain

Samples of utility functions

- The utility of an additional copy of a message at any time
- It solely reflects network conditions
- For a highly loaded network (complex function):

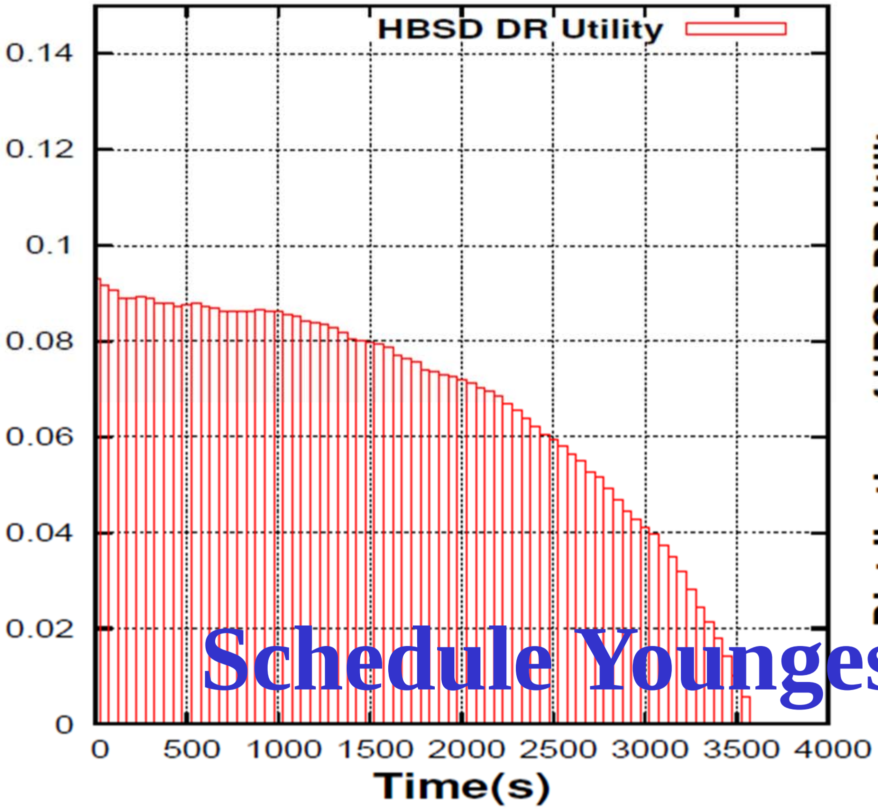


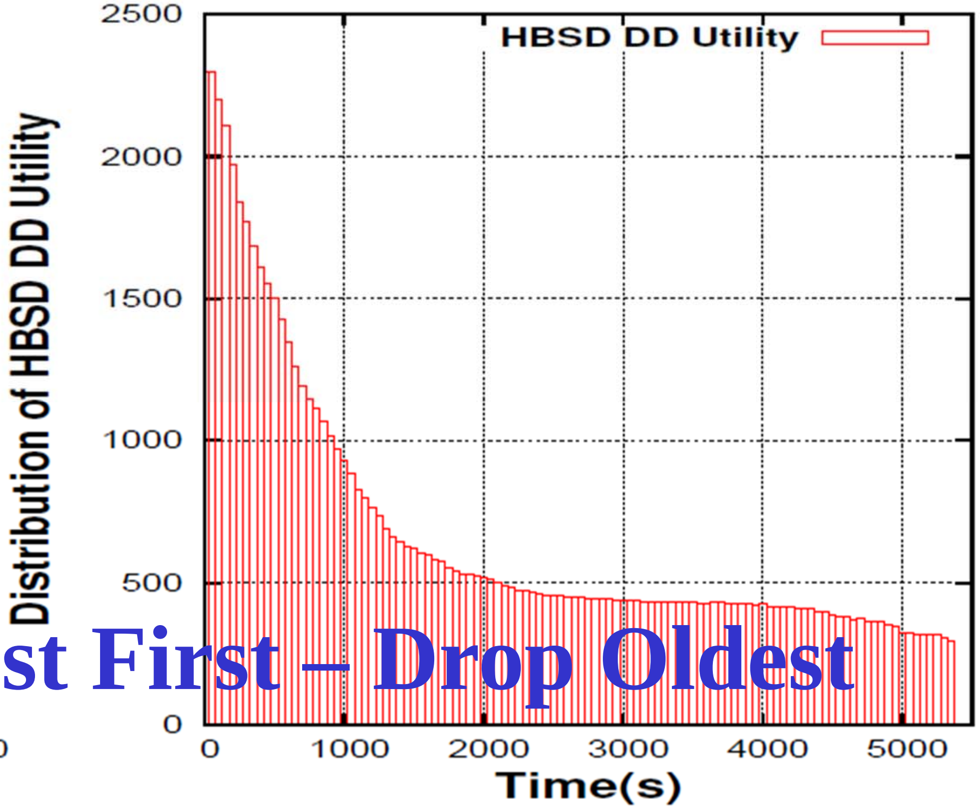


INRIA SOPHIA ANTIPOLIS

Samples of utility functions

- For a lightly loaded network, it seems things are easier and simple policies can be applied:

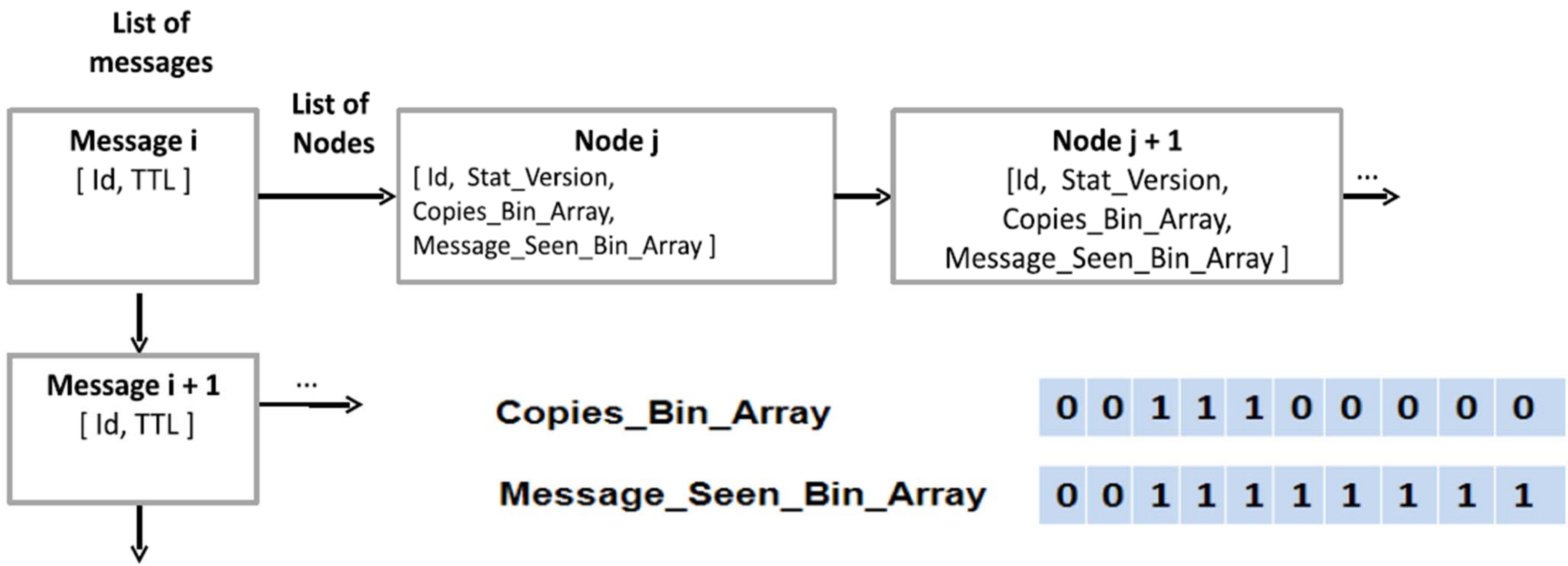




Schedule Youngest First – Drop Oldest

Reducing the signaling load

- To build the history, nodes need info on past messages:
  - How many copies each message had in the network
  - And this is for each moment of its life (we bin the lifetime)



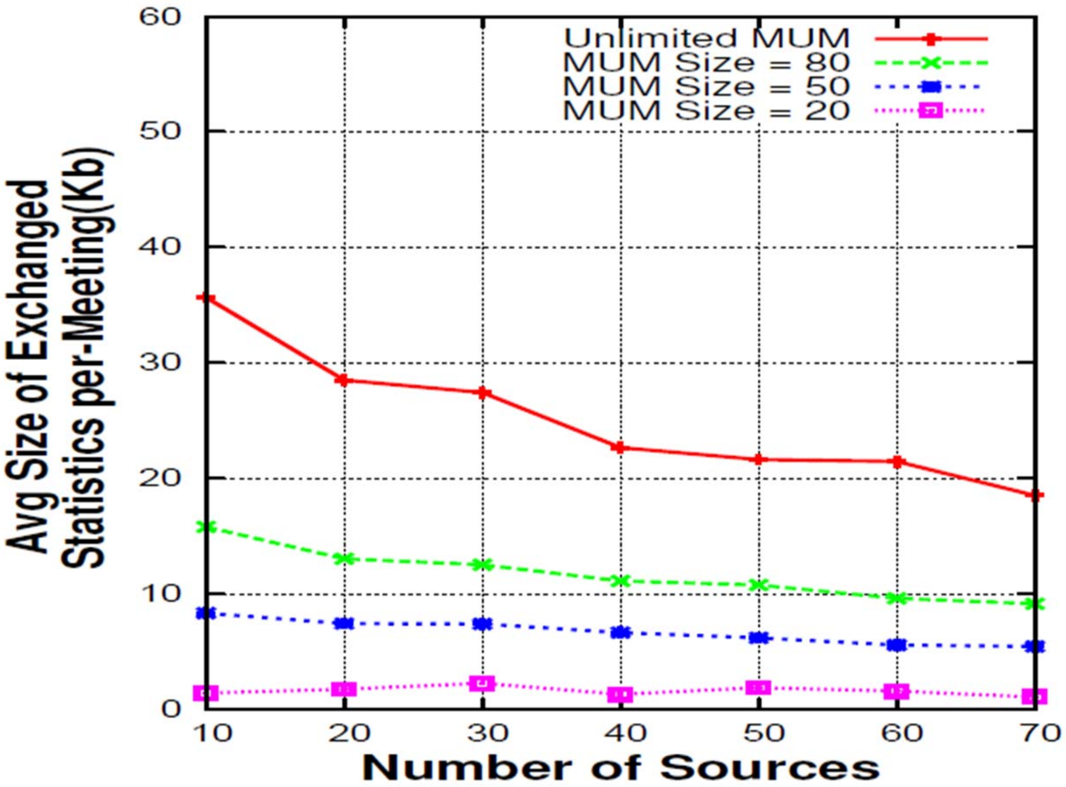
Reducing the signaling load

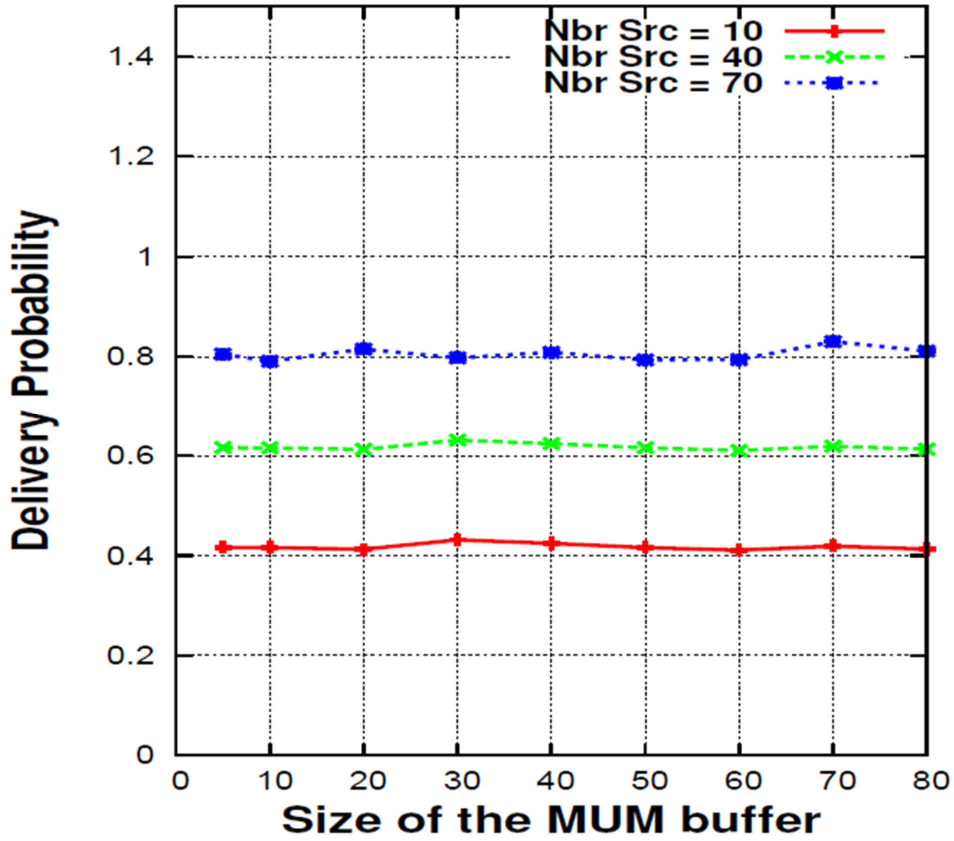
- Several optimizations:
  - Binning: Increase time granularity and hence reduce information
  - Message sampling: A node selects some messages and track them while moving (who got them and when)

    MUM: Number of Messages Under Monitoring
  - Updates' filtering: Nodes exchange information when they have something new to say on a message (new copy and drop)
- When a message TTL expires, move it to the history cache
- Signaling traffic can be limited to few KBytes per contact

Reducing the signaling load

- Volume of signaling traffic





The signaling traffic even decreases with the load !!
(the more the load the less the new events / message)

No pay in performance

Implementation / Web page

- HBSD is available for the network simulator NS2
- And is also available for the DTN2 architecture as an external router (in C++)
- Code has been recently tested in the Scorpion testbed at the University of California Santa Cruz
- Code, papers, presentations are available at:

http://planete.inria.fr/HBSD_DTN2/

Conclusions

- An analytical framework to better understand the scheduling and drop of messages in DTNs
- Two optimal policies for the cases of delivery rate and delay
- A distributed version of the optimal policies that WORK
- Validation with a synthetic mobility model and real traces

- Next steps:
  - Study the interaction with routing protocols
  - Extension to a publish/subscribe scenario (> 1 dest)

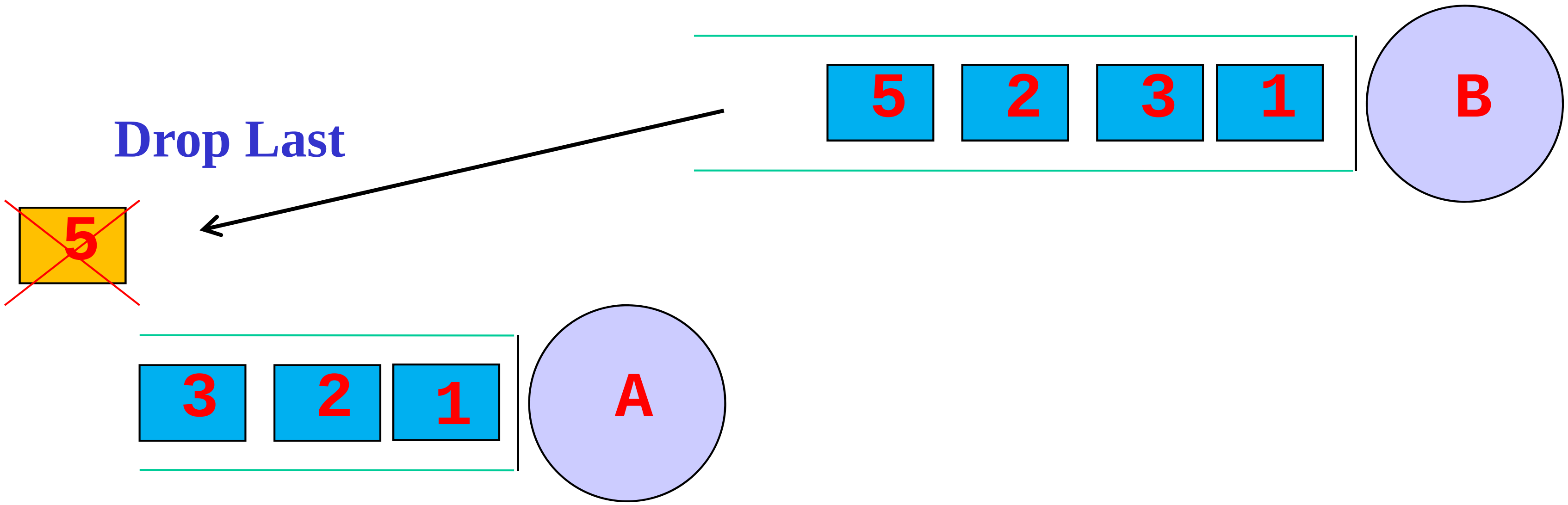
Thanks for your attendance !

Email: Chadi.Barakat@sophia.inria.fr

WEB: http://www.inria.fr/planete/chadi

Message Drop in DTNs

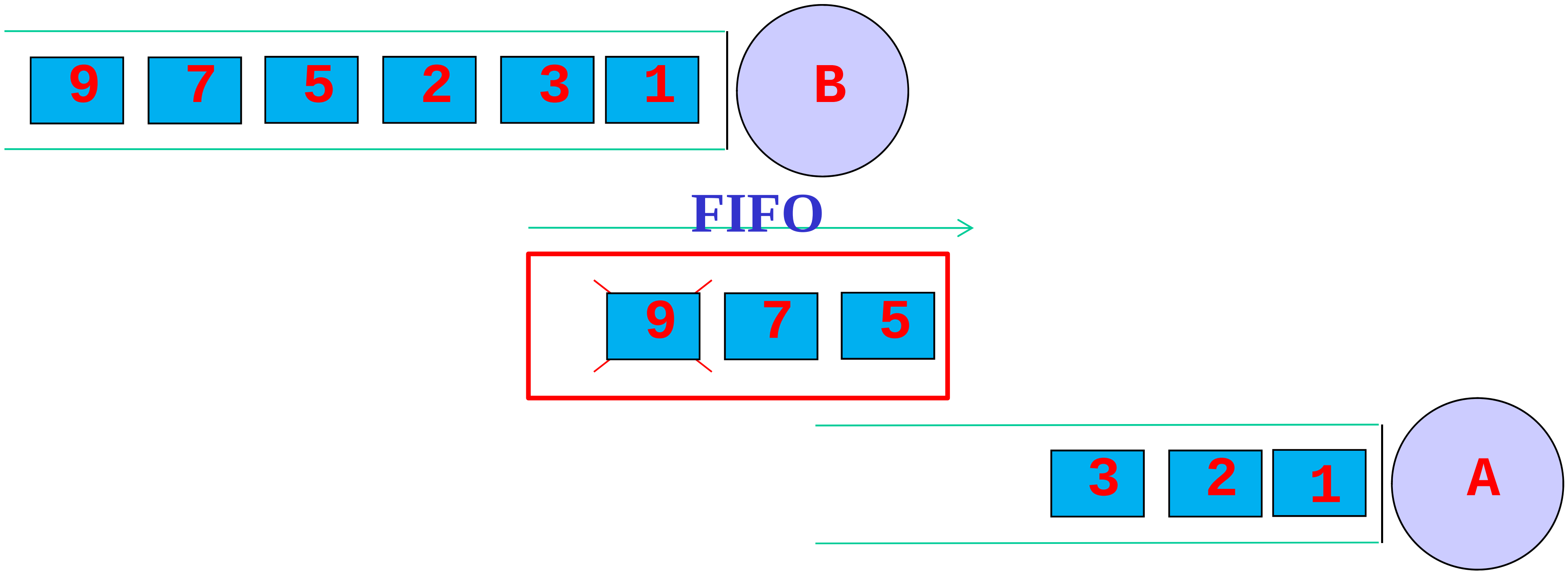
B gives A a copy of message 5 but A's buffer is full



Could be a bad decision if message 5 is a young message

Message scheduling in DTNs

B gives A the three missing messages in FIFO order, but A leaves before 9 is sent



Could have been better to give message 9 first