

Opportunistic DTN Routing with Window-aware Adaptive Replication

Gabriel Sandulescu and Simin Nadjm-Tehrani

Computer Science and Communications Research Unit
University of Luxembourg

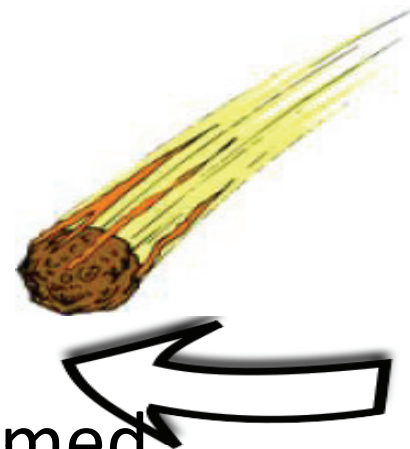
LASSY – 02/07/2008

Talk overview

- Delay Tolerant Networks context
 - Assumptions
 - Scenarios & constraints
 - Routing
- ORWAR – **O**ppportunistic **R**outing with **W**indow-aware **A**daptive **R**eplication
 - main ideas
 - evaluation

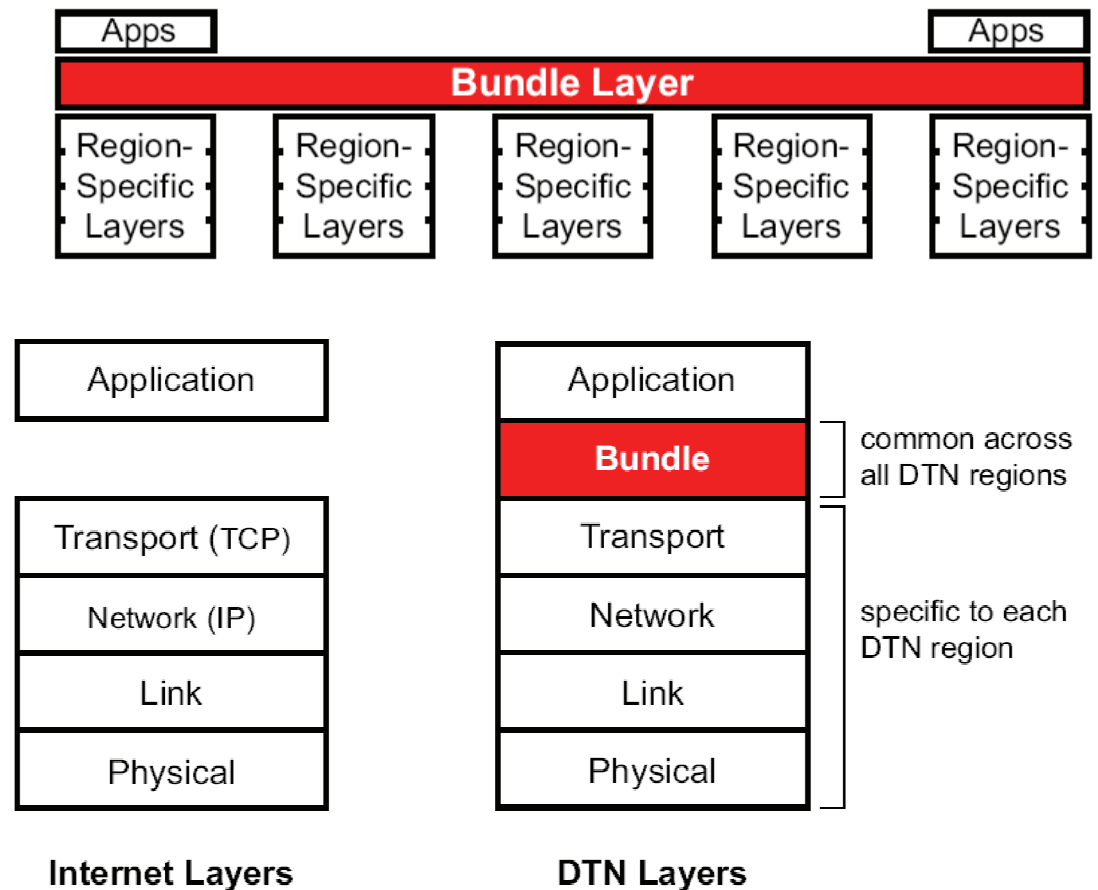
Delay- and disruption-tolerant networks

- Beginnings - 1998
- Assumptions
 - End to end path is not assumed
 - Frequent disruptions
 - Long and uncertain delays
- Aims
 - Achieving reliable communication under stressed and unreliable conditions



DTN Specifics (1)

- A network of regional networks
- Heterogeneous Transport Layer: not only TCP



Source: F. Whartman, A DTN Tutorial

DTN Specifics (2)

- Low density, message replication and custodians
- Contact:
 - Scheduled
 - Opportunistic
- Non-conversional protocols (data and metadata bundled). Bundles can be LARGE !

Typical scenarios

- Interplanetary networks
 - high latency links, satellite obstruction, scheduled
- Sensor networks
 - extreme energy constraints
- Pocket switched networks
 - heterogeneity
- Military wireless networks
 - need important scalability

Routing, soft taxonomy

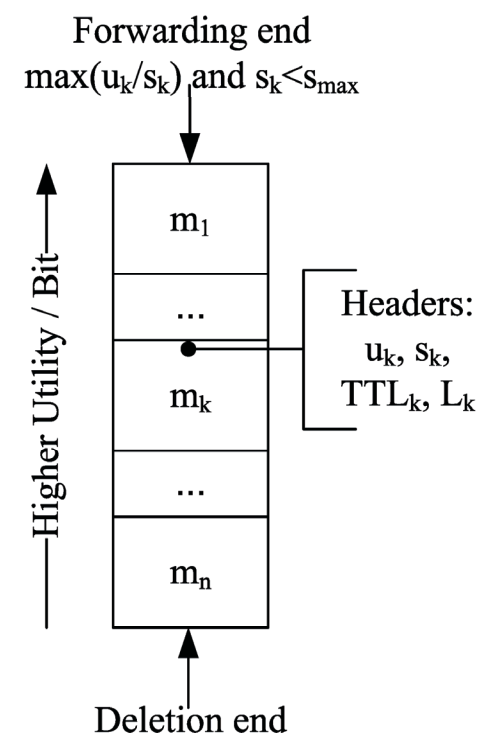
- How much knowledge about the network:
 - Oracle $\leftarrow \rightarrow$ Pure Opportunistic
 - In between knowledge about contacts or knowledge about message load
 - ORWAR – pure opportunistic
- How much replication
 - Epidemic $\leftarrow \rightarrow$ Single copy
 - In between – fixed number of copies (ORWAR)

ORWAR Main ideas

- Use local knowledge (speed, radio range) for estimating **contact window** and the **biggest transferable message**
 - Diminishing partial transmissions
 - Energy efficiency
- Resource allocation:
 - More replicas for bigger utility messages
 - **utility/bit** as selection criteria for forwarding and message deletion

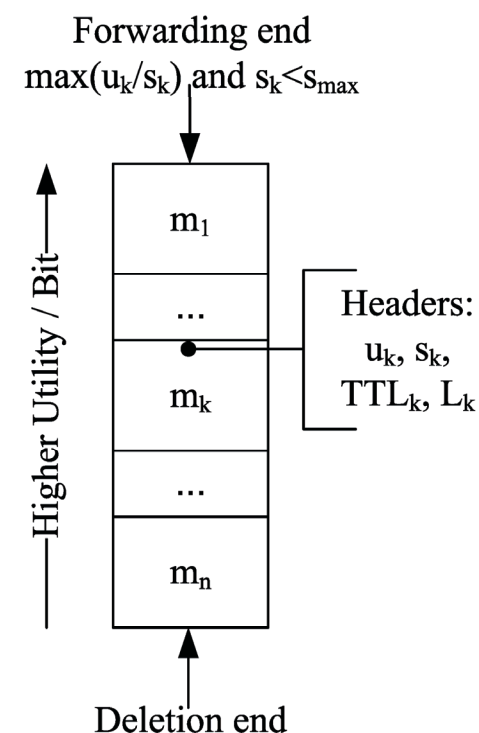
Protocol specifics (1)

- Compute the size of maximal exchangeable message for current contact window (s_{\max})
- Messages forwarded starting with the best utility/bit (u_k/s_k) not exceeding size: s_{\max}
- Deletions: reverse order



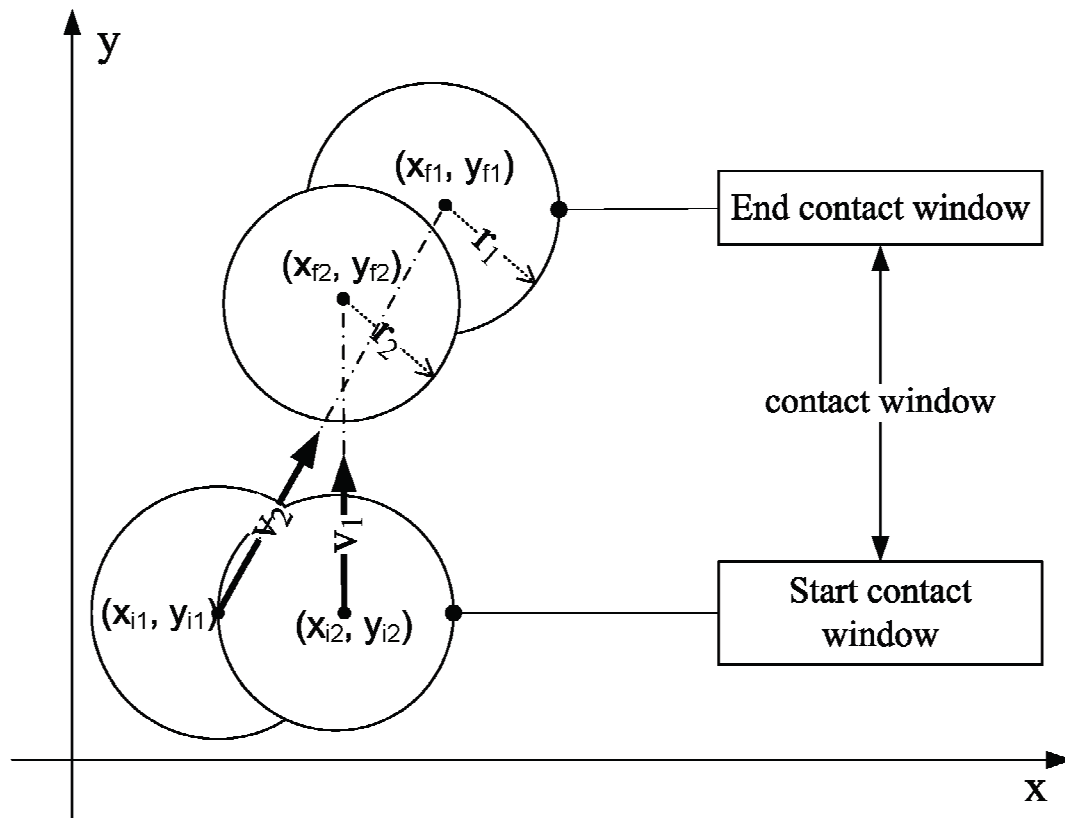
Protocol specifics (2)

- L_k : number of initial replicas,
 - dependent on message utility
- Bundles known to be delivered are removed
- This implementation: Maps DTN priorities to utility



Contact window estimation

- Speed, location, radio range is transmitted during meeting
- Contact lasts as long as $d < \min(r_1, r_2)$
- s_{\max} is computed before message forwarding

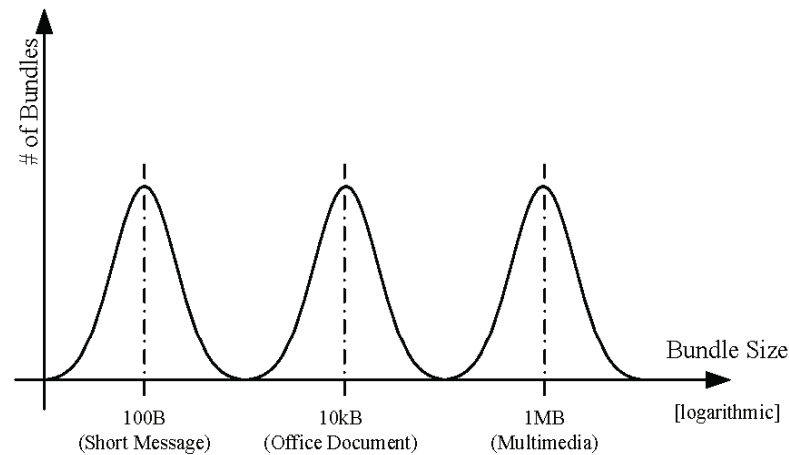


Evaluation setup

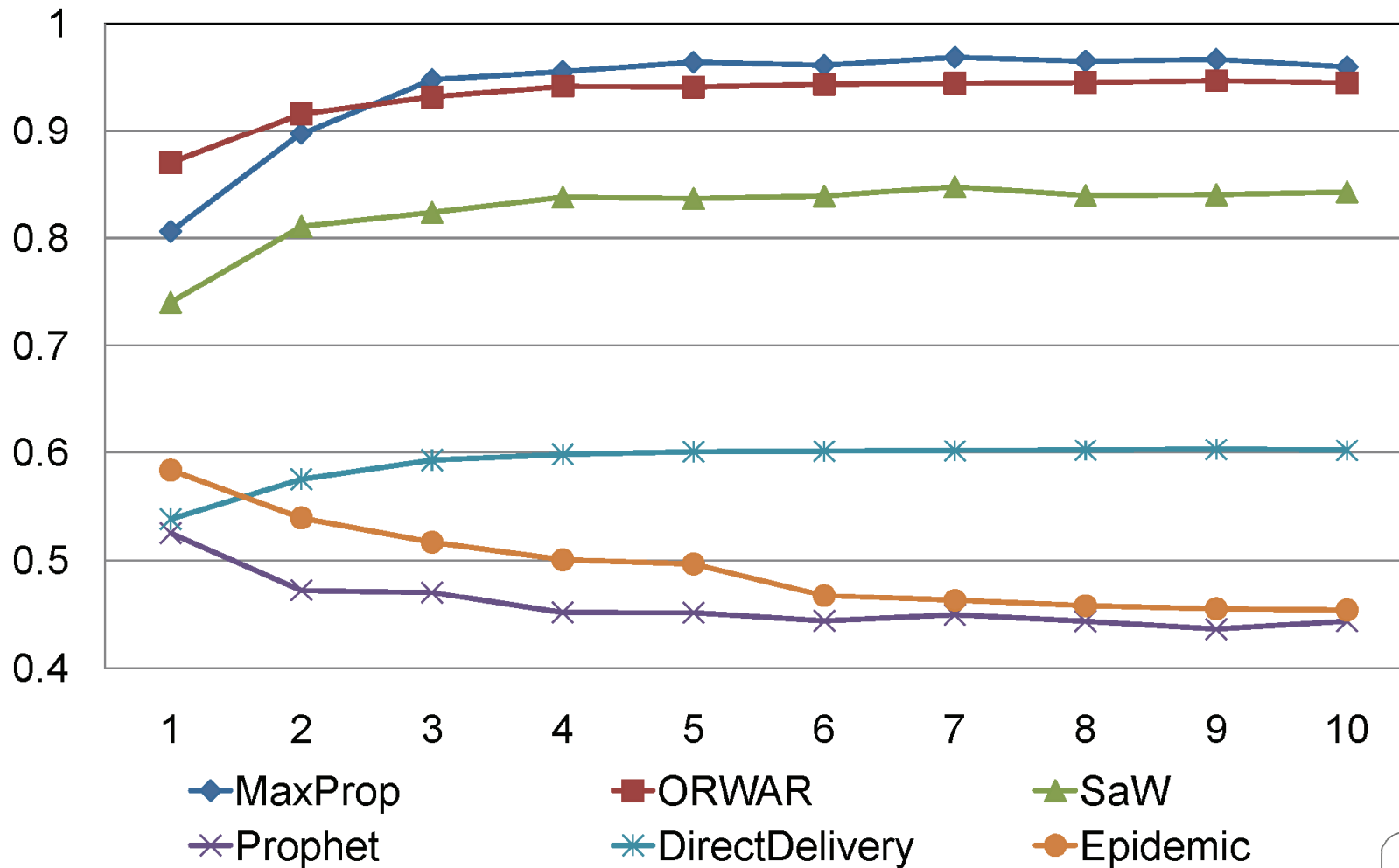
- Using ONE (Opportunistic Network Environment)
- 126 nodes (80 pedestrians, 40 cars, 6 trams) in a 4500m x 3500m playground
- Mobility as in ONE (Map based movements)
- Transmission range 10m for pedestrians, 20m for cars and trams transmission speed 250 kBps

Traffic characteristics

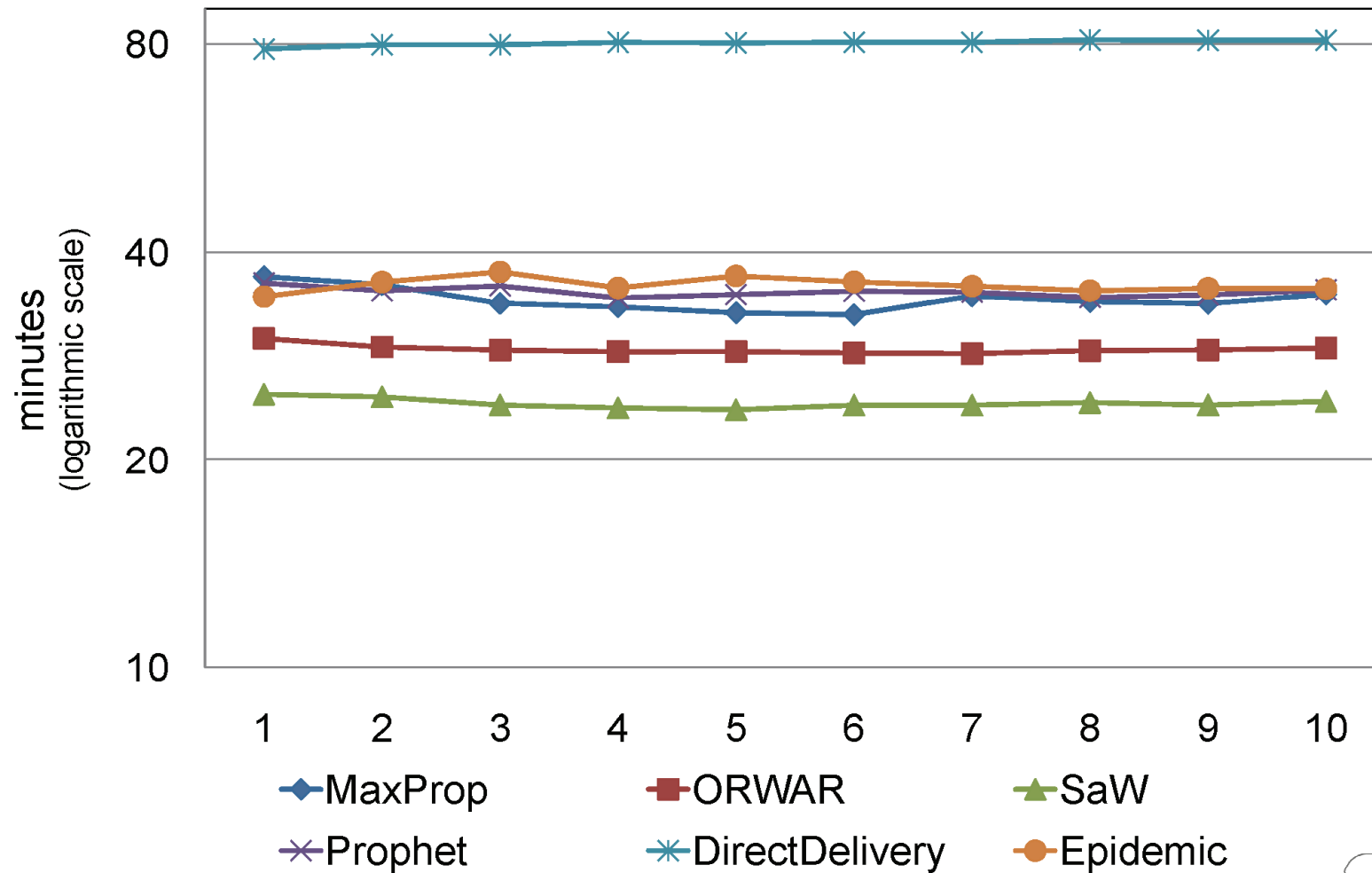
- Injecting 3000 messages in 12h
- Bundle size initially distributed as below
- Within each 3 utility levels, distributed evenly between messages



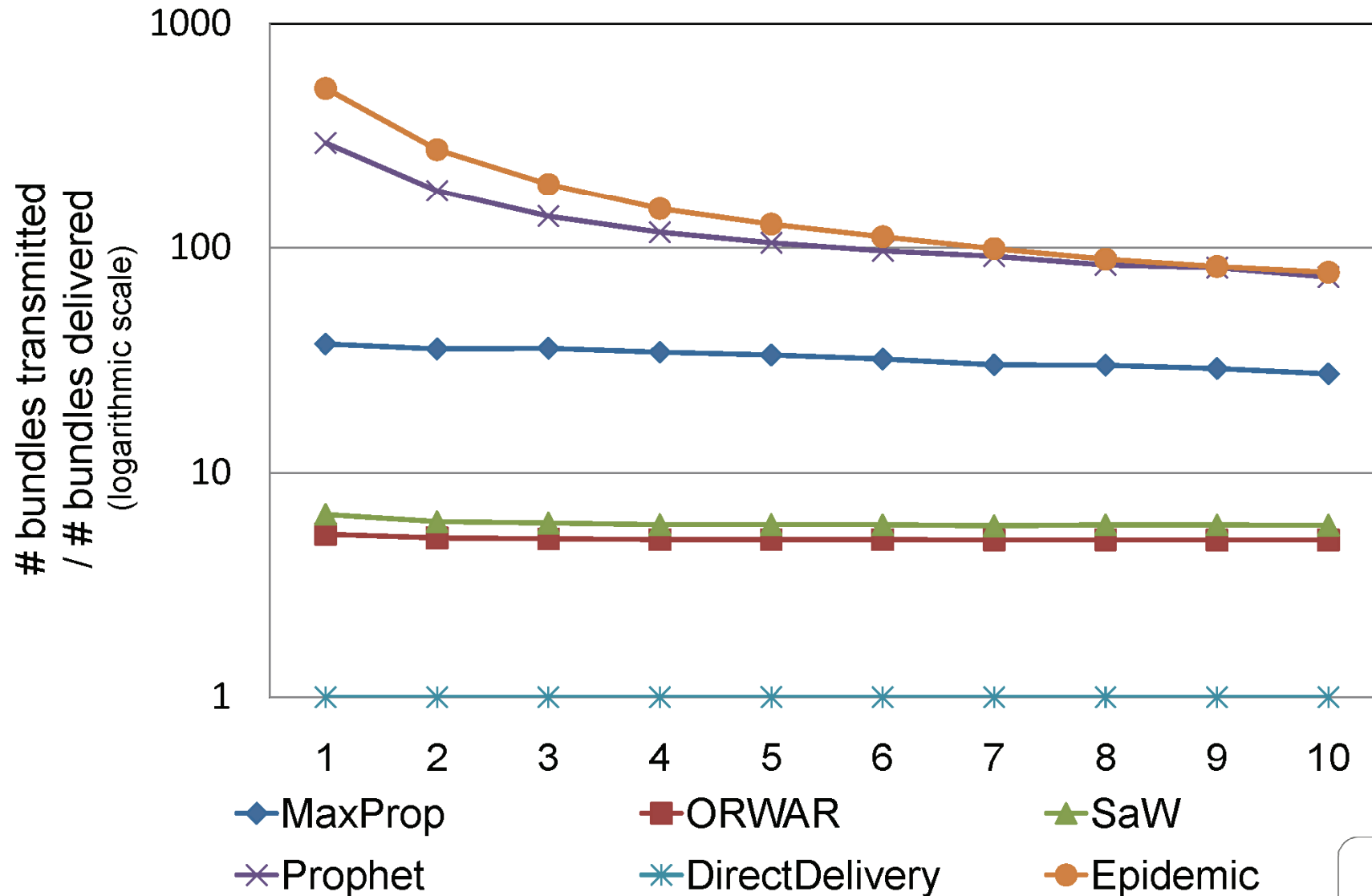
Message size effect on delivery ratio



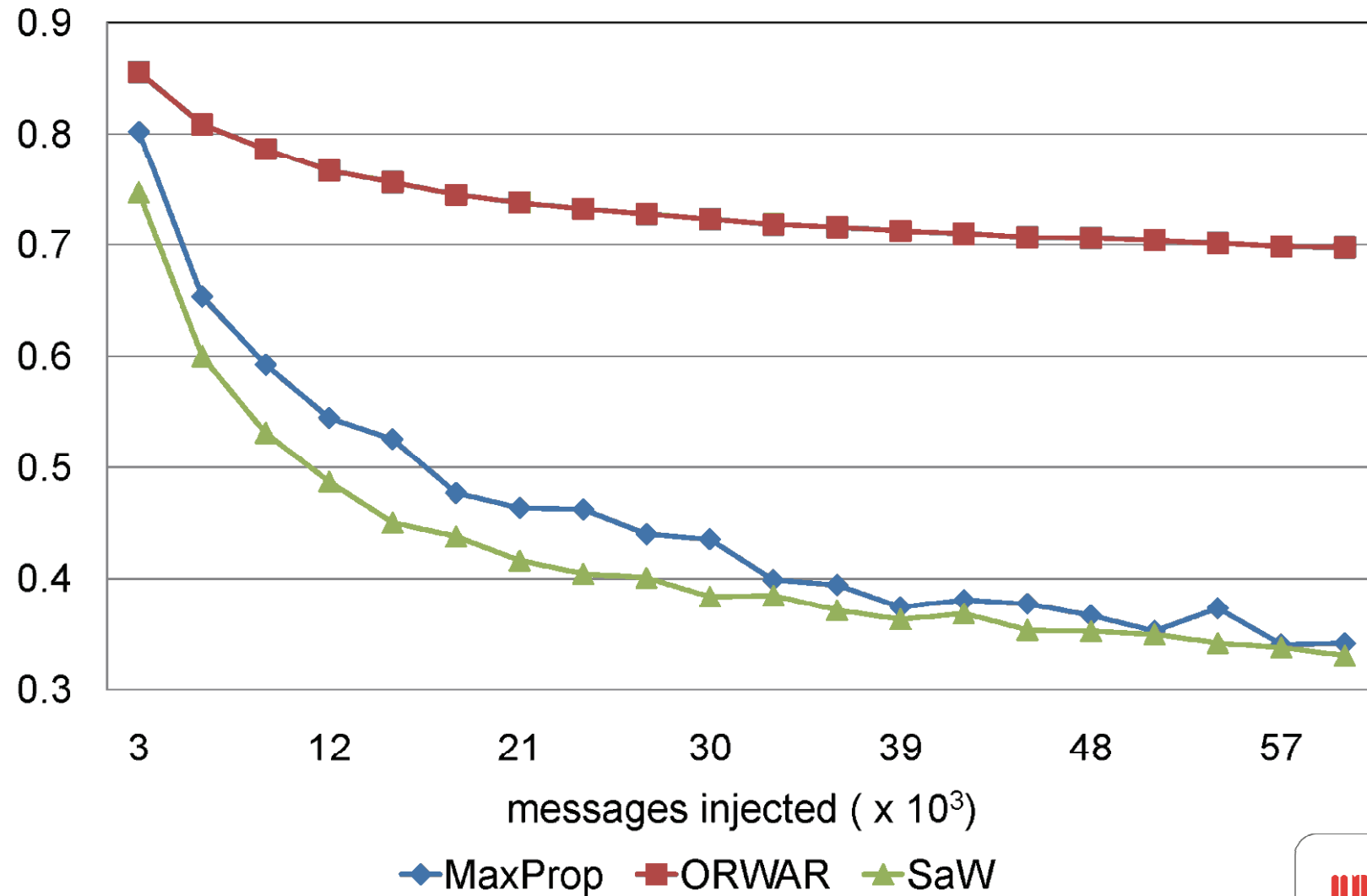
Message size effect on latency



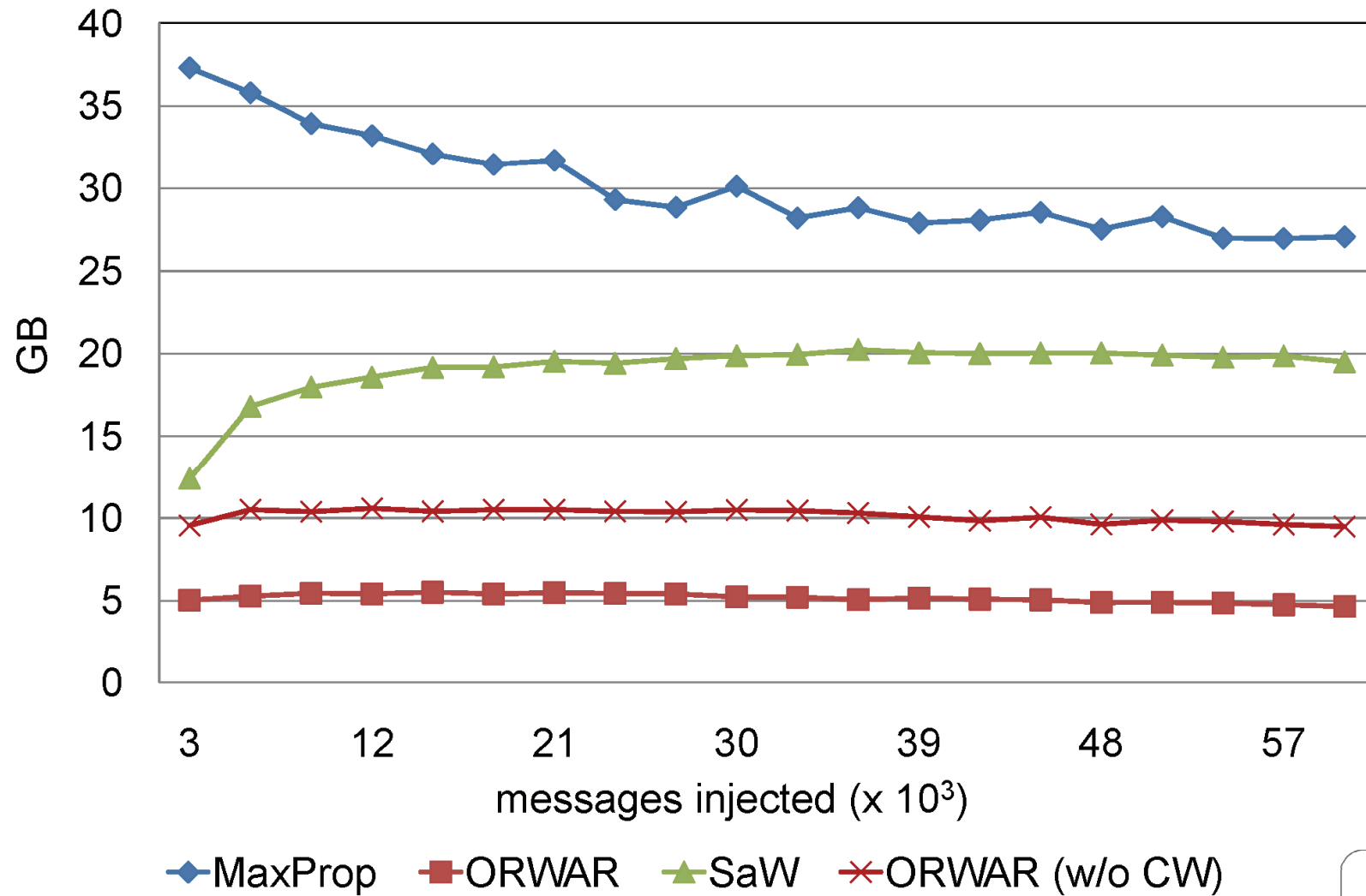
Message size effect on overhead



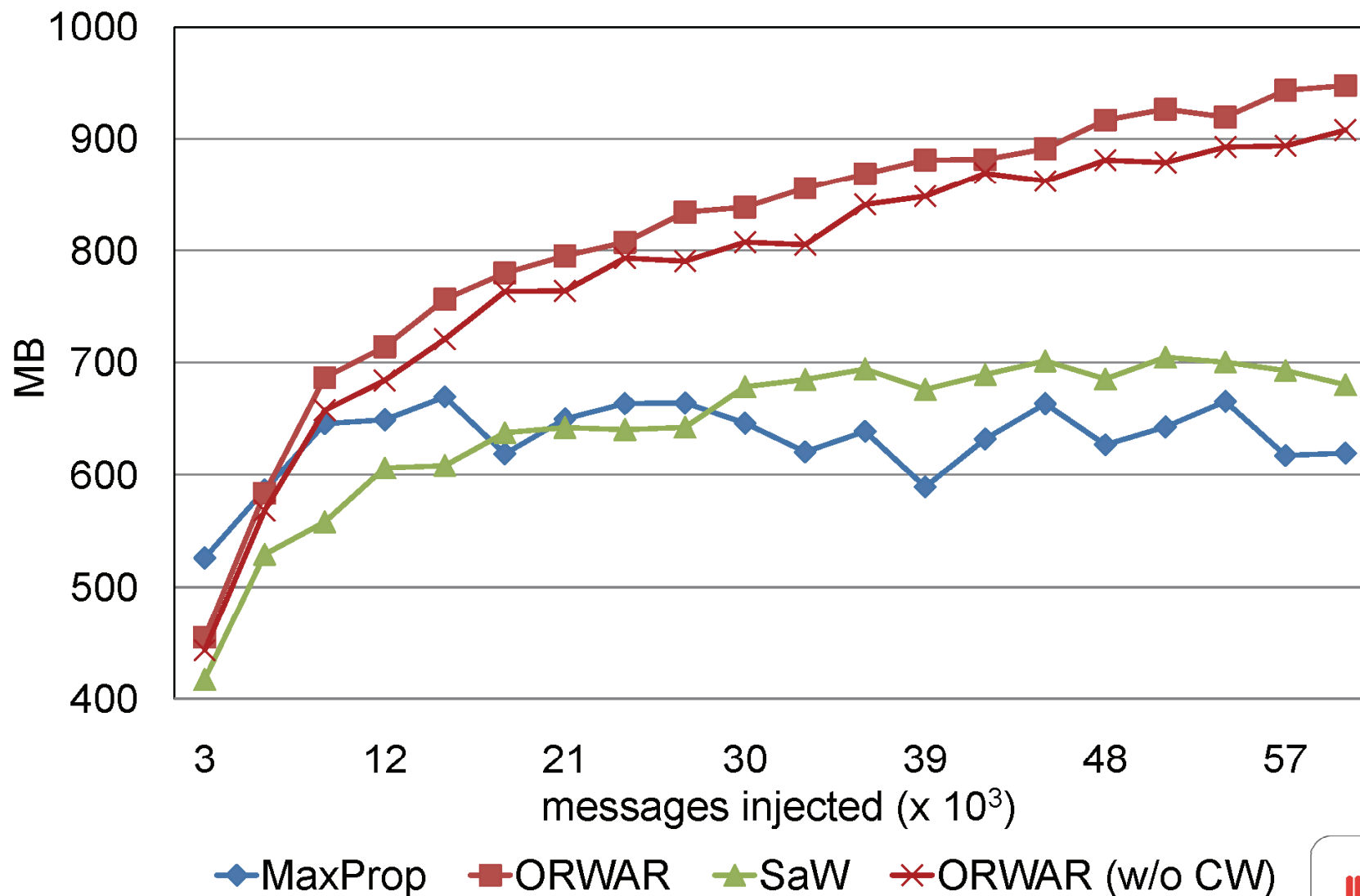
Delivery rate versus load



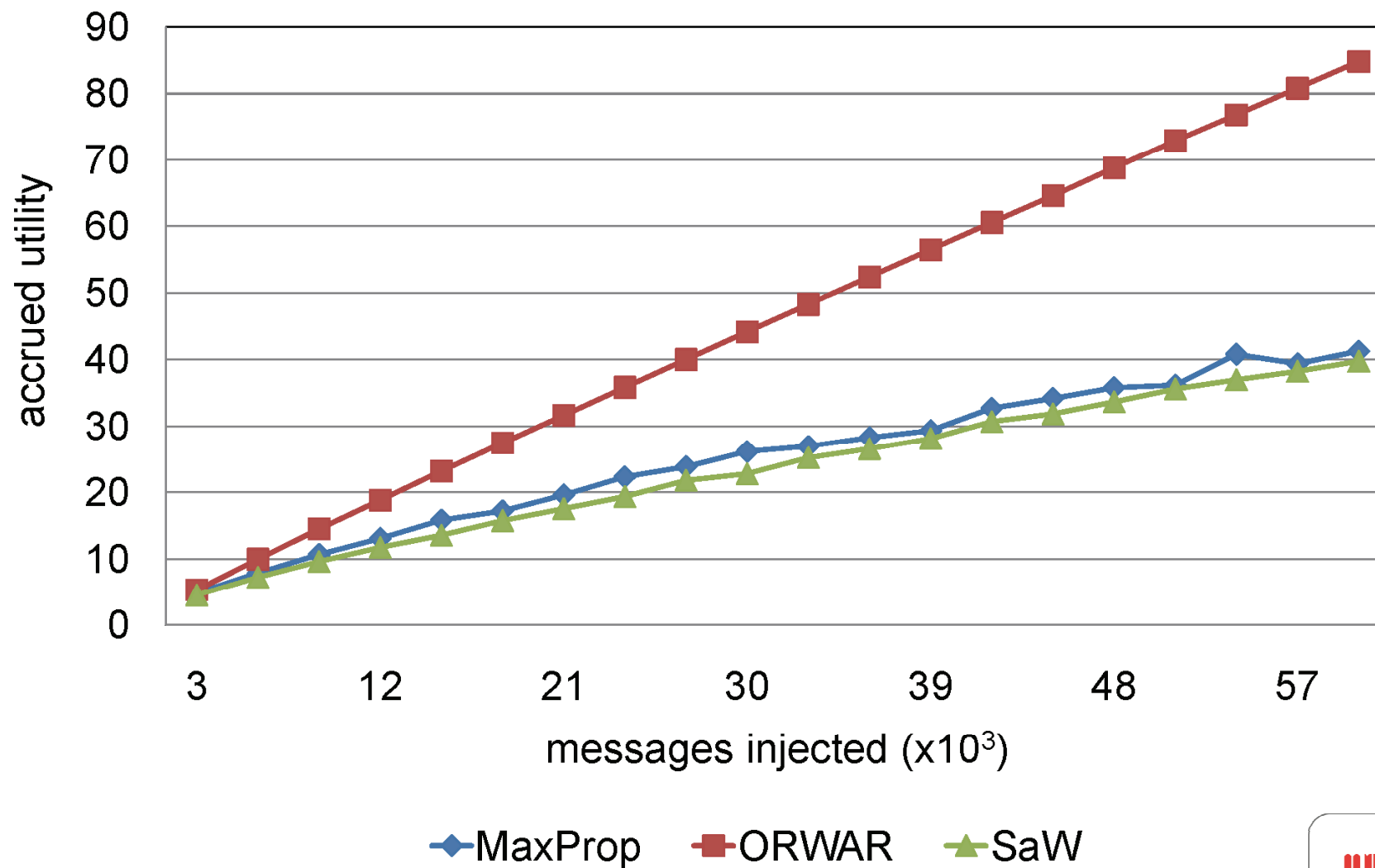
Partial transmissions versus load



Throughput vs. load



Accrued utility vs. load



Conclusions (1)

- Superior performance compared with five existing algorithms (*Direct Delivery, Epidemic, Prophet, MaxProp and SprayAndWait*)
- At medium load ORWAR has a similar delivery rate to MaxProp while creating only 20% of its overhead
- At high load ORWAR has the best delivery rate and best overhead

Conclusions (2)

- Suited for big sized messages
- Can be used where fragmentation is not available
- Accumulated utility can be used as a new metric

Future work

- Adding adaptive proactive fragmentation: taking the most valuable message and fragment this up to s_{\max}
- Supposing utility not constant in time. Utility in function of remaining TTL
- Emulation instead simulation

THANK YOU !