EPIDEMIC OVERLAYS FOR MULTICAST IN MOBILE AD HOC NETWORKS

Masters Thesis

Michael Sirivianos

11/05/2004

Overview

- Problem Definition-Related Work (Gossip)
- Proposed Solution-Contributions (Deterministic flooding).
- Ad Hoc Routing Protocols
- Membership Management
- Per-hop Expander Graph Creation
- Application Layer (Overlay-based) Deterministic Flooding
- Experimentation. Comparative Evaluations.
- Discussion and Conclusions
- Extensions and Future Work.

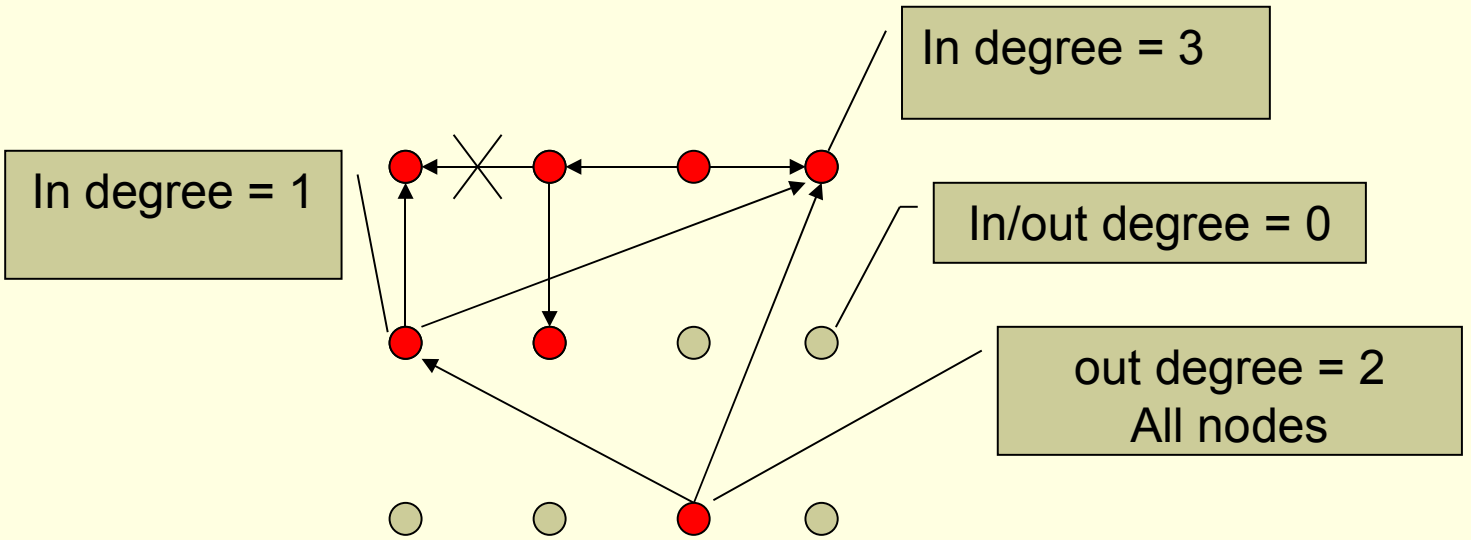
Problem Definition-Existing Solutions

- Problem: Scalable and reliable multicast in ad-hoc networks.
- Existing solutions:
  - In-network: Tree-based and mesh-based proposed solutions. *ODMRP, MAODV, CALM, RALM.*

    Application Layer: AMROUTE, CAMP *etc*.
  - Application layer gossip-based multicast. *RDG*.
  - Per-hop gossip. Relay frame to next hop with a given probability. *Gossip based Ad-hoc Routing* [Sasson et al].

Problem Definition-Gossip

- The non-deterministic nature of ad hoc networks provides incentives for epidemiological (gossip-based) protocols.
- Assumption: Any two nodes can send messages to each other.



Problem Definition-Gossip

- Application Layer Gossip Protocols
  - Pros
    - Scalable. No need to maintain state that describes expensive tree or mesh structures, at each node.

      Need to maintain view state (partial, probabilistically acquired e.g LPB).
    - Reliable and failure resilient through message redundancy.
    - Adaptable. No need to reconfigure multicast structures in the presence of node/link failures. Low cost join/leaves.

Problem Definition-Gossip

- Application Layer Gossip Protocols
  - Cons
    - High message overhead. Scarce network resources not optimally utilized.
    - No reliability guarantees (random graph connectivity). The out-degree (fanout) is balanced, but the in-degree may be highly unbalanced.
    - Proposed membership management (e.g LPB, SCAMP) not suitable for ad hoc networks. Knowing a node does not mean we can resolve a route to him at a low cost.

Proposed Solution - Deterministic flooding

- Gossip Vs Deterministic Flooding [Lin, Marzullo, Masini 00]
  - Impose a k-connected, link minimal Harary overlay geometry. Flood deterministically over it.
  - Comparing to pure gossip protocols, it induces lower message overhead (lower fan-out). It provides high reliability guarantees (increased connectivity) and graceful degradation.

Proposed Solution-Random Regular Graphs

- Random k-regular graphs: Random graphs in which each node has exactly *k* adjacent nodes.
- Good expanders.
- Desirable properties for communication systems design.
  - O(log(*N(logN)))* diameter [Bollobas & de la Vega 1982]
  - Remains highly connected following random removal of linear-size subsets of edges or nodes [Goerdt 01, Alon et al. 02]
  - *a.a.s k*-connected, if 3 < k < |V|^(0.02) [T.Luczak 92]

Araneola

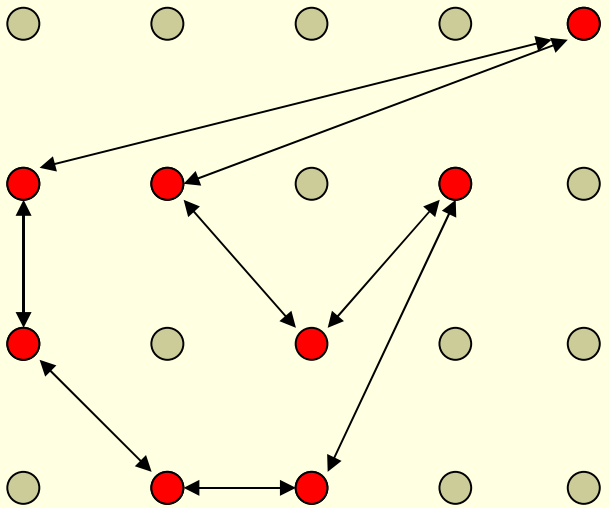
- Distributed algorithm that probabilistically creates an overlay over which data are deterministically disseminated.
  - Dynamic maintenance with constant cost for join/leave.
  - The degree of every node converges to either *k* or *k* + 1. 90% k.
  - Achieves the three mathematical properties of *k*-regular random graphs

Araneola

- Araneola combines the best of two worlds:
  - the resource utilization efficiency and reliability of deterministic overlay-based routing
    - High connectivity guarantees with lower fan-out. Balanced in-degree/out-degree.
  - the scalability, failure resilience and adaptability of epidemiological protocols.
    - Fully decentralized. Partial network knowledge per member.
    - Low cost join/leaves.
    - Path redundancy-> graceful degradation in the presence of failures.

Araneola's Overlay

Portion of a 2-regular overlay:



Contributions

- Our main contribution:
  - Investigate applicability of probabilistic k-regular overlay creation and deterministic flooding over it, in ad-hoc multi-hop topologies.
- Per-hop: using broadcast to nodes in range (neighbors).
  - View consists of physical neighbors. Only connected nodes process broadcast messages.
  - Compare to brute force flooding and meta-data based flooding.

Contributions

- Rely on an ad-hoc routing abstraction to cope with limited connectivity.
  - Determine most suitable tunnelling mechanism-Optimize it.
  - Incorporate RDG membership tracking in Araneola. Route driven view using reactive DSR.
  - Rely on link-state FSR. LPB-like view.

Contributions

- Make Araneola protocol more efficient and aware of route state and mobility.
  - Determine and optimize most suitable reactive or proactive unicast routing protocol.
  - Mobility adaptation, topology awareness, routing layer promiscuous mode and other cross-layer optimizations that port Araneola to the ad hoc environment.
- Compare with Route Driven Gossip, and per-hop flooding variants.

Related Work

- Gossip in wired networks
  - LPBcast [Eugster et al]
    - Gossips uniformly about data packets, message digests and membership state, providing reliability without imposing a complete membership view -> Scalable
- Gossip in ad-hoc networks
  - Anonymous Gossip
    - enhances MAODV by gossiping message requests from to multicast group members. Routes to gossip destination resolved during multicast route discovery.
  - Route Driven Gossip [Patrick, Eugster et al]
    - route driven view acquisition. Augments DSR with view acquisition primitives. View depends on route state.

Related Work

- Gossip Based Ad hoc Routing.
  - Gossip: a node broadcasts a received message with probability p instead of flooding.
  - To support unicast routing protocols that utilize flooding such as AODV.
  - Investigates bimodal behavior of gossip configurations. In almost all executions the message hardly reaches any nodes or all nodes receive the message. In accordance with results found in percolation theory

Related Work

- Distributed Construction of Expander Graphs

  [Yeung, Siu. 2003]
  - Randomized distributed algorithm for constructing 2d-regular graphs consisting of d Hamilton cycles.
  - With high probability $\log_d N$ diameter.
  - A node joins in $\log_d N$ time(rounds) with logdN messages
  - A node leaves in O(1) time and messages.
  - As opposed to Araneola where nodes join and leave with 3k messages
  - Assumptions: Every node can communicate with each other. No volatile multi-hop topologies considered.

Ad Hoc Routing Protocols

- Common attribute the MAC Layer.
  - 802.11 RTS/CTS for hidden terminal. Per-frame ACKs.
- Some **proactive** routing protocols
  - Fisheye State Routing (Scoped Link State).
  - Optimized Link State Routing (Link State)
  - Wireless Routing Protocol (Distance Vector).
- Some **reactive** Routing Protocols
  - AODV-uses destination route reply sequence numbers.
  - DSR- source routing.
- Cluster-based
  - Zone routing protocol (Hybrid: proactive for intra-zone, reactive for inter-zone)

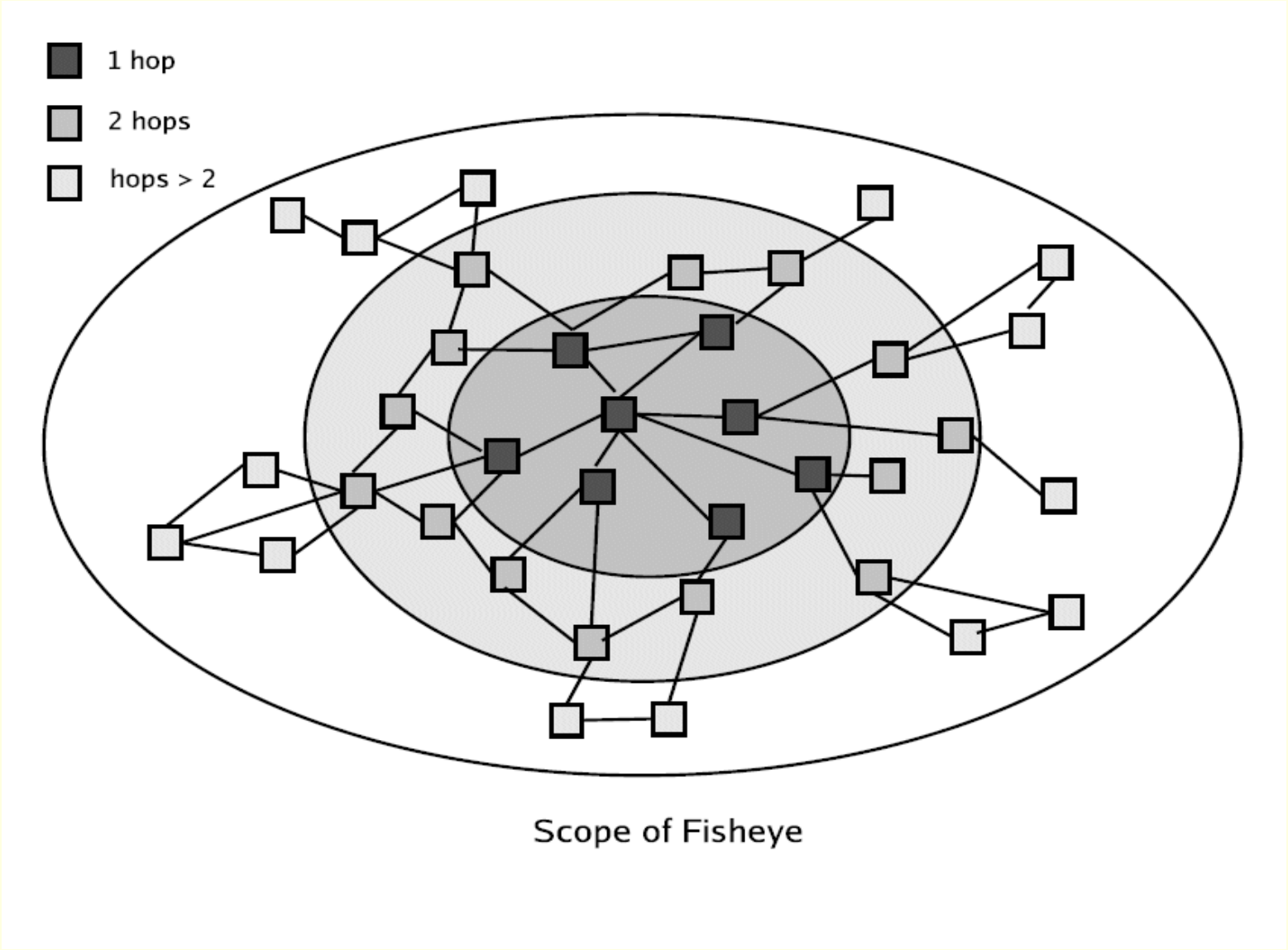
Fisheye State Routing

- Link state protocol
- Topology map is maintained at each node
- Normally FSR would have to periodically transmit link state packets.
  - This floods the network with control packets.
  - Limits scalability.
- FSR circumvents this:
  - Link state packets are no longer flooded
  - Only neighbors exchange topology network information
  - Link state exchange is time and not event triggered

Fisheye State Routing

- FSR uses different exchange intervals for different types of entries in the topology table
- Link state entries within scope are propagated to neighbors more frequently
- What is a scope?

Fisheye State Routing



AODV

- AODV is an on-demand reactive protocol inspired by DSDV
- Hop by hop forwarding.
- RREQ is initiated until it reaches the destination or a node with a “fresh enough’’ route.
- Destination Sequence numbers issued by dests are employed to:
  - ensure loop free routing
  - Utilize the most recent routing information.
  - To reply to a RREQ nodes should maintain entries regarding the destination that have DS numbers greater than or equal to that contained in the RREQ.

AODV

- During forwarding of RREQ reverse paths are established
- RREP is unicast back to the originator of the RREQ
- Nodes in the path, set up forward routing entries that point to the node from which the RREP was received
- Hello messages are used to detect link failures
- Each node maintains a forwarding table with the next hop for each destination and its sequence number.
- A list of predecessor nodes per routing table entry is maintained as well.
  - Predecessors are notified upon link breakage.

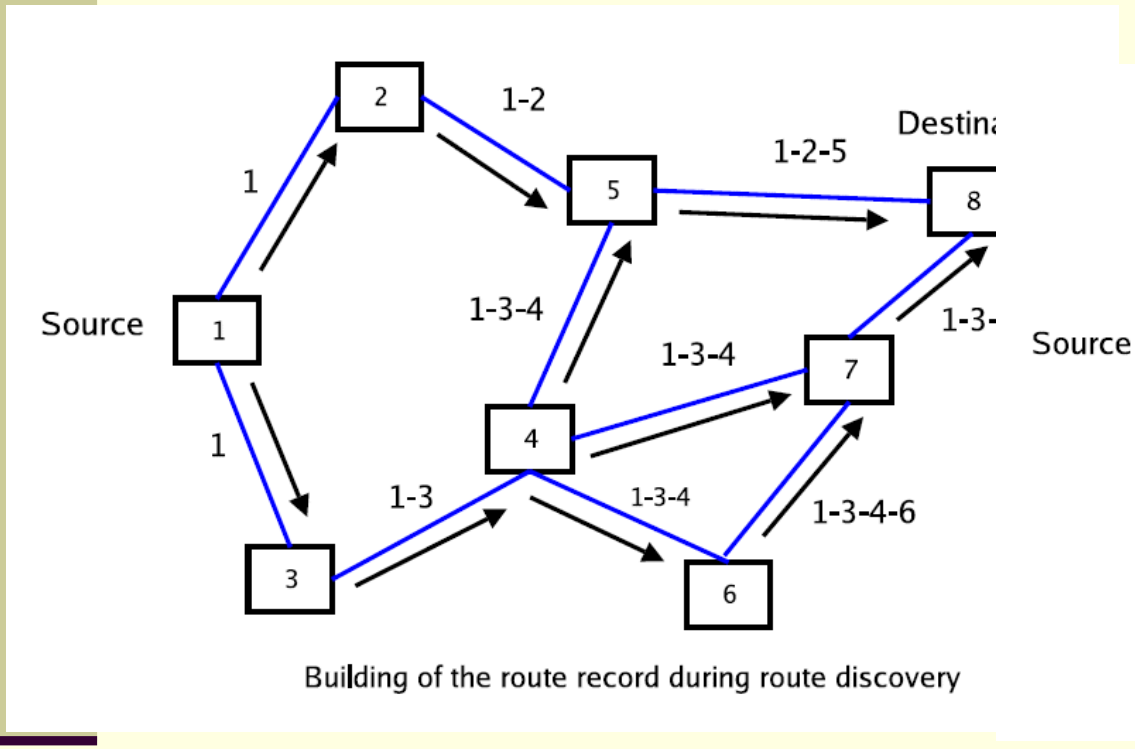
Dynamic Source Routing

- Reactive source routing protocol.
- Aggressive route caching
- Route Requests (RREQ) are initiated when a route to the destination does not exist in cache
- Route Replies (RREP) are generated by intermediate routes if route to destination available or by the destination itself
- They are source routed to the RREQ initiating node
- Intermediate nodes learn forward routes

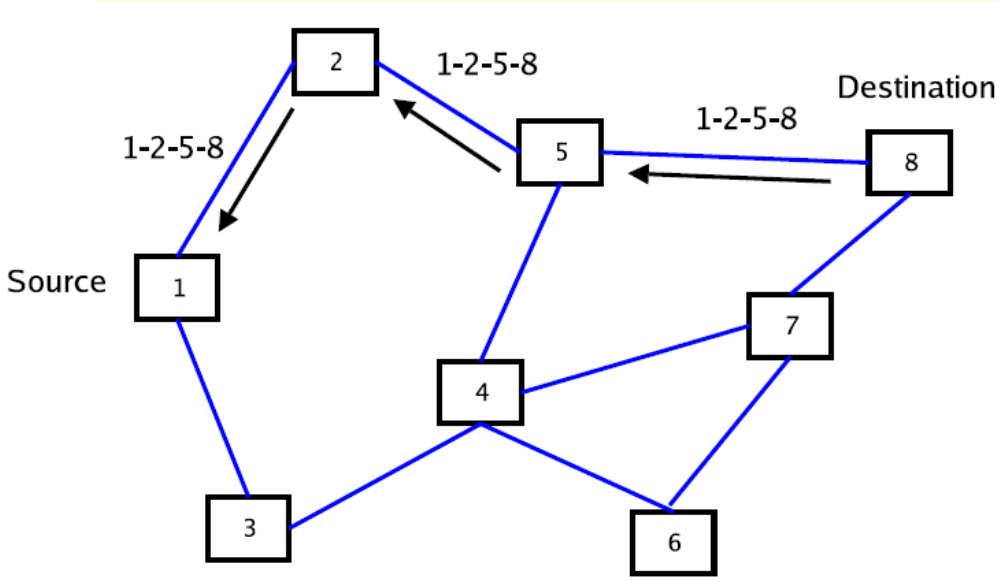
Dynamic Source Routing- Built-in Optimizations

- Promiscuous learning of source routes
  - Listen to packets that are not unicast to itself. Read the paths and acquire new routes through the previous hop of the packet.
- Optimum route selection based on hop count metric
- Processing of all route messages by all nodes
- Replying from cache by intermediate nodes
- Gratuitous route replies
  - Promiscuously acquire a packet that includes itself in the path. Remove entries between the previous hop and itself from the path. Send shorter RREP to packet source.
- Salvaging
  - if RRER choose retrieve alternative route and re-route packet.

Dynamic Source Routing



Building of the route record during route discovery



Propagation of the route reply with the route record

RREQ Flooding

RREP Source Routing

Dynamic Source Routing-Opt

- Lack of effective mechanism for expiring stale routes
  - Design flaw aggravated by aggressive route caching
  - Low reliability in presence of frequent topological changes

- Optimizations:
  - lifetime-based Route Cache eviction policy. Route cache TTL optimized to decrease routing delay.
  - Route discovery with backward learning. Request receivers learn the reverse path.
  - Check for duplicates when reply for cache.

Route Cache Eviction Policy

- A method for calculate route cache lifetime to optimize routing delay was introduced by Lyang and Haas. in INFOCOM ‘03.

- Besides reducing buffering requirements, it aims at decreasing the number of stale routes in route cache, thereby improving reliability.

Route Cache Eviction Policy

- Some definitions:
  - Routing delay R : time to resolve a route to a given destination
  - T is the route TTL
  - t is the time that passed since route was cached and a new packet is sent to the destination
  - L is latency per link, D is number of hops
  - q(t) is the probability a link to still be up after time t
  - fdata(t) the pdf of the time t between data packet transmissions. Beyond scope.

Route Cache Eviction Policy

- If t > T then the route is not valid and the routing delay is 2LD
- If t < T then there are two possible cases
  - In the first case, an intermediate node at hop count i detects broken link and reports it back, thus

    $$R = 2Li+2LD$$

  - In the second case, the route is correct and there is no routing delay

    $$R = 0$$

Route Cache Eviction Policy

$$
\begin{aligned}
R_{t \le T}(t,T) &= 2L \sum_{1}^{D} i[1-q(t)]q(t)^{i-1} + 2LD[(1-q(t)] \\
&= 2L[D + \frac{q^D(t)-1}{q(t)-1} - 2Dq^D(t)] \\
R_{t>T} &= 0 \qquad (1)
\end{aligned}
$$

- Mean R is calculated according to the probabilities of the various events and then averaged over t in (0, ∞) using fdata(t)

$$\overline{R}) = \int_0^\infty R f_{data}(t)\, dt$$

Route Cache Eviction Policy III

- With derivation of mean R by T to determine the minimum, it is shown that q(Topt) is the root in [0, 1) of g(x).

$$g(x) = 2Dx^D - \frac{x^D - 1}{x - 1}$$

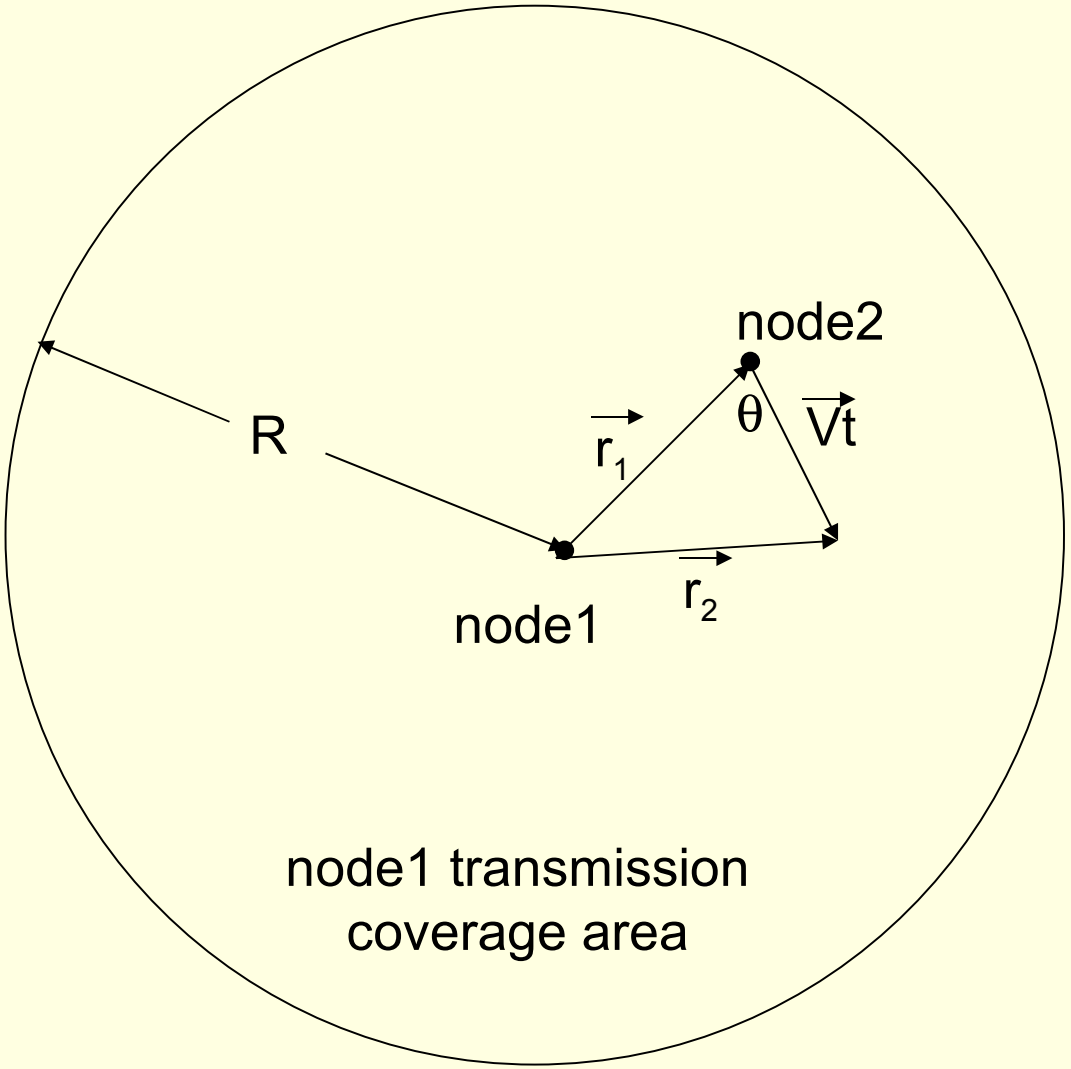
- Link-up times are exponentially distributed with mean time μu.

$$q(T_{opt}) = 1 - \int_0^{T_{opt}} \frac{1}{\mu_u} e^{-\frac{t}{\mu_u}}\, dt = e^{\frac{T_{opt}}{\mu_u}}$$

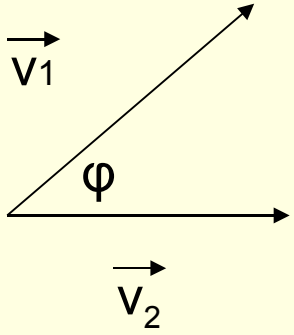
$$T_{opt} = -\mu_u \log q(T_{opt})$$

- What is the value of μu? They set it equal to an arbitrary value (1 s). We approximate it as follows.

Route Cache Eviction Policy IV



Velocity
vectors



Route Cache Eviction Policy

- Average relative speed v. Problem reduced into having only one mobile node.

$$v = \frac{1}{2\pi} \int_0^{2\pi} \sqrt{(v_1 \cos\phi - v_2)^2 + (v_1 \sin\phi)^2}\, d\phi$$

- Initial distance between nodes uniformly distributed from 0 and R.

$$\overrightarrow{r_2} = \overrightarrow{r_1} + \overrightarrow{v} \times t$$

$$|\overrightarrow{r_2}| = \sqrt{|\overrightarrow{r_1}^2| + v^2 \times t^2 - 2v \times t|\overrightarrow{r_1}| \times \cos\theta}$$

$$t = \frac{r_1 \times \cos\theta \pm \sqrt{-r_1^2 \times \sin\theta^2 + R^2}}{v}$$

Route Cache Eviction Policy

- Below we derive the expected time for two connected nodes to fall out of each other's transmission range. Assuming v1 = v2, v = 4v1/π.

$$E[t] = \int_0^{2\pi} \int_0^{R} \frac{r_1 \times \cos\theta + \sqrt{r_1^2 \times \cos\theta^2 - r_1^2 + R^2}}{2\pi \times R \times v} \, dr_1 \, d\theta.$$

$$E[t] = \int_0^{2\pi} \int_0^{1} \frac{\pi R(r \times \cos\theta + \sqrt{1 - r^2 \times \sin\theta^2})}{4v_1} \, dr \, d\theta.$$

- The above integration is approximated with numerical integration yielding

$$\mu_u = E[t] = 5.7416 \frac{\pi R}{4v_1}$$

LPB-Membership Tracking

- LPBcast membership algorithm:
  - Gossip Message contains:
    - (a) Notifications (b) Notification IDs (c) Unsubscriptions (d) subscriptions
  - Procedures:
    - (a) Gossip Reception (b) Gossip Sending
      (c) Subscribing (d) Unsubscribing
  - Subs/Unsubs update view. Hosts gossip to F randomly selected members of the view.
  - View dissemination coupled with notification dissemination.
  - Enhance independence and uniformity in view distribution, assigning weights to views.

LPB

- Theoretical analysis.
  - LPB
    - View size l, does not affect the notification infection rate.
    - The rate depends on fan out, F. View does impose F<= l limit

Araneola Protocol Description

- Protocol description
  - Data Structures
  - Connect Task
  - Disconnect Task
  - Gossip Task
  - Secondary Tasks: a) Failure Detection b) Garbage Collection

Araneola Protocol Description

Data structures:

id – this node's identifier.
neighbors – set of triples ⟨id,degree,heartbeat⟩, initially ∅.
messages – queue of messages tagged with *m.id*, initially ∅.
missing_msg – set of identifiers, initially ∅.
heard_from:missing_msgs ⟶ list of nodes.
recent_mids – set of messages identifiers, initially ∅.
clock – the current time.
last_connect_to_time – a time.

Constants:

LOW – target number of neighbors.
HIGH – upper bound on the number of neighbors.
MIN_HEARTBEAT – failure detection parameter.
Timeouts: connect_timeout, failure_detection_timeout, disconnect_timeout, connect_to_timeout, gossip_round_timeout.

Araneola Protocol Description

Connect Task:

```
1.  loop forever
2.      gap ← LOW − |neighbors|
3.      while gap > 0
4.          n ← random node
5.          send ⟨CONNECT, |neighbors|⟩ to n
6.          gap ← gap − 1
7.      sleep (connect_timeout)
```

Failure Detection:

```
22. loop forever
23.     sleep (failure_detection_timeout)
24.     foreach n ∈ neighbors
25.         if n.heartbeat < MIN_HEARTBEAT then
26.             send LEAVE to n
27.             remove_connection(n)
28.         else
29.             n.heartbeat ← 0
```

Araneola Protocol Description

Event Handlers-Procedures:

```
8.  upon receive ⟨CONNECT, d⟩ from n do
9.      if |neighbors| < HIGH then
10.         add_connection (n, d,true)
11.     else
12.         send ⟨REDIRECT,lowest degree neighbor⟩ to n

13. upon receive ⟨REDIRECT, n′⟩ from n do
14.     send ⟨CONNECT, |neighbors|⟩ to n′
```

```
15. upon receive ⟨CONNECT_OK, d⟩ from n do
16.     if |neighbors| < HIGH then
17.         add_connection (n, d,false)
18.     else
19.         send ⟨LEAVE⟩ to n

20. upon receive ⟨LEAVE⟩ from n do
21.     remove_connection(n)
```

```
Procedure add_connection (node_id n, int d, boolean ack)
30. neighbors ← neighbors ⋃ {n, d,MIN_HEARTBEAT}
31. if ack = true then
32.     send ⟨CONNECT_OK, |neighbors|⟩ to n

Procedure remove_connection (node n)
33. remove n from neighbors
34. remove n from all heard_from lists
35. if |neighbors| < LOW then
36.     wake up connect task
```

Araneola Protocol Description

Disconnect Task:

```
1.  loop forever
2.      sleep (disconnect_timeout)
3.      i ← |neighbors| − LOW
4.      if i > 0 then
            /* Rule 1 */
5.          cands ← set of i neighbors with highest degrees
6.          foreach c ∈ cands
7.              if c.degree ≤ LOW then
8.                  cands ← cands \ {c}
9.              else if c.id < id then
10.                 send ⟨DISCONNECT⟩ to c
            /* Rule 2 */
11.         if cands = ∅ then
12.             h ← neighbor with highest degree
13.             l ← neighbor with lowest degree
14.             if |neighbors| ≥ l.degree + 2 then
15.                 send ⟨CONNECT_TO,h⟩ to l
```

Araneola Protocol Description

Event Handlers:

```
16. upon receive ⟨DISCONNECT⟩ from n do
17.     if |neighbors| > LOW then
18.         send ⟨DISCONNECT_OK⟩ to n
19.         remove_connection(n)

20. upon receive ⟨DISCONNECT_OK⟩ from n do
21.     remove_connection(n)

22. upon receive ⟨CONNECT_TO, n′⟩ from n do
23.     if |neighbors| ≤ LOW ∧
24.         clock − last_connect_to_time > connect_to_timeout then
25.         send ⟨CHANGE_CONNECTION,|neighbors|, n⟩ to n′
26.         last_connect_to_time←clock

27. upon receive ⟨CHANGE_CONNECTION,d, n′⟩ from n do
28.     if |neighbors| < HIGH then
29.         add_connection (n, d,true)
30.         send ⟨DISCONNECT⟩ to n′
```

Araneola Protocol Description

Gossip Task:

```
1.  loop forever
2.      sleep (gossip_round_timeout)
        /* Send gossip messages to neighbors */
3.      foreach n ∈ neighbors
4.          create new gossip message m, with new m.id
5.          m.degree ← |neighbors|
6.          m.ids ← {recent_mids | source ≠ n}
7.          m.reqs ← ∅
8.          foreach mid ∈ missing_msgs
9.              if heard_from(mid).first = n then
10.                 m.reqs ← reqs ∪ {mid}
11.         send ⟨GOSSIP,m⟩ to n
        /* Update data structures */
12.     move 1st element of each heard_from(mid) list to end
13.     recent_mids ← ∅
```

Araneola Protocol Description

Event Handlers:

```
14. upon receive ⟨GOSSIP,m⟩ from n do
15.     neighbor(n).heartbeat ← neighbor(n).heartbeat + 1
16.     neighbor(n).degree ← m.degree
17.     foreach id ∈ m.ids ∧ id ∉ messages
18.         missing_msgs ← missing_msgs ∪{id}
19.         append n to heard_from(id)
        /* Send requested messages to n */
20.     foreach r ∈ reqs
21.         send ⟨DATA,message with identifier = r.id⟩ to n

22. upon receive ⟨DATA,m⟩ from n do
23.     messages.enqueue(m)
24.     missing_msgs.remove(m.id)
25.     recent_mids ← recent_mids ∪{m.id}
```

Per-hop Regular Graph Creation (PHA)

- View consists of nodes within transmission range.
  - Periodically transmit neighbor beacons
  - Check heartbeat to determine, if a node is still neighbor

- Efficient medium utilization.
  - Connect, Disconnect, Gossip messages are a bundle that contains the intended recipients to minimize header overhead.
  - Gossip message contains list of recent IDs and list of (request ID, heard_from) pairs.
  - Data task.
    - Read gossip for requests with heardFrom==myAddress
    - Add requests in request queue. Each request queue entry contains a list of nodes that requested the message.
    - Periodically scan request queue and create a data packet that contains the actual message and the requesting recipients. Reset queue.

PHA Downsides

- Does not adapt to connectivity changes.
- All nodes process received messages at the application layer.
- (1 to 1.5)log(N) neighbors required for connectivity [Xue, Kumar 2004] .
  - 100 nodes network-> ~7 nodes.
  - 1000m x 1000m, 50 nodes, random waypoint, neighbor beacons every 2 s, 2 m/s -> 6,86 avg neighbors
  - Need to establish connections to all neighbors. Limited connectivity.
  - More suitable for dense networks.

PHA Downsides

- Broadcasts bundles → no collision detection and avoidance mechanism → loss of data and control messages.
- Essentially, gossiping of message IDs, and data pulling from advertizing neighbors, a version of controlled flooding.
  - For not very dense networks overlay maintenance task is redundant.
- In sparse graphs, is brute-force flooding or meta-data-based flooding more efficient?

Controlled Flooding

- Gossip task. Periodically broadcasts to all next hop neighbors recent ids and request (message ID, heard_from) pairs all bundled in a gossip packet.
- Gossip handler determines missing messages and adds requests in a queue to be served by the data task.
- Data task. Periodically broadcasts requested data messages.
- A simple distributed coordination mechanism
  - because only the node which first received and advertised a data message broadcasts it, alleviating contention.

App Layer Deterministic Flooding

- Building an Araneola-based application layer deterministic flooding scheme.
  - Use a unicast protocol to provide abstraction of higher connectivity. Expected to increase the size of a membership view. Thus, leverage the randomness of the regular overlay geometry, and thereby its "good" properties.
  - Route information that is obtained reactively is valuable.
  - flood-based route discovery is expensive.
- There exist no connectivity guarantees for every pair of nodes in the network.
  - A random membership tracking mechanism, which is aware of the route state is more appropriate than LPB's view-driven approach. It would not trigger route requests for every destination being selected as an overlay neighbor.

Route Driven Gossip

- Can use any on-demand routing protocol's route discovery phase for view acquisition. Associates view with route state.
- Data Structures
  - Active, Passive, Remove View and

    Data Buffers(new,old)
- Join Session
  - Group Requests. Send Group Reply with *p* if in same group.
- Gossip/Leave Session
  - Gossip messages carry new data, random members and gossip-pull requests.

Route Driven Araneola (RDA)

- Incorporation of RDG view mechanism in Araneola. RDA view partitioned in Active, Passive and Remove views. DSR augmented with Group Request Group Reply messages. DSR/RDA interface:
  - *DSR-RDA_InitiateGroupRequest()* – upon joining
  - *DSR-RDA_GetHopCount()* – for topology awareness
  - *DSR-RDA_RouteExists()* – for view management
  - *RDA-DSR_RouteUpdateNotification()* –Notify RDA upon route state changes. For view management.
  - *RDA-DSR_PromiscuouslyPeekMessage()* – For routing layer promiscuous operation

Route Driven Araneola

- Modifications of base-Araneola
  - Connect Task:
    - Select connection candidates from Active View.
    - Upon reception of connect message add sender in Active view.
  - Gossip Task:
    - Transmit 4 messages per data packet.
  - Disconnect Task:
    - Do not remove corresponding heard_from entries when connection to a member is removed. Different failure model.

Route Driven Araneola

- Route Update Notification Policy.
  Three choices:
  - DSR always notifies RDA upon route acquisition regardless of how it is acquired. Induces locality of view information.
  - It notifies RDA only upon the reception of Group Requests/Replies. View tracking is driven by Group Requests and the gossip task. Routes are still aggressively acquired. A periodic task checks route state and places existing view members to view partitions accordingly.
  - It notifies RDA only upon reception of Group Requests/Replies. Routes are acquired only through this mechanism. (No aggressive routing). View is not promptly populated and is not updated regularly

Route Driven Araneola

- Membership and overlay neighborhood tracking policy.
  - First policy, Add Gossip Sender in Active view:

The receiver of a *gossip* from a non-overlay neighbor does not ignore it. If the receiver's degree does not exceed the high threshold, then it adds the message source to its neighbor set, adds the source to Active view, regardless of view state. Otherwise, the receiver sends a *leave* message to the source.

When a node receives a redirect or connect_to or change_connection message it checks if its degree exceeds the high threshold. If not, it adds the indicated for connection nodes in the Active view, regardless of route state, and initiates connection.

Note that with this policy, member ID's can reside in Active view without a route to the corresponding members being known.

Route Driven Araneola

- Membership and overlay neighborhood tracking policy.
  - Second policy, Connect to Reachable only:

The *gossip* receiver adds the source in Active or Passive view according to reachability. If sender is reachable, and degree is below high threshold, connection is added, otherwise a leave message is sent to the sender. If sender is not reachable, then the message is ignored.

The *redirect* receiver, checks the route state of the indicated node. If it is reachable, it sends a *connect* message to the specified node and augments the active view. Otherwise, it initiates connection to randomly selected active node, and if it is already in its membership view it adds the indicated node to the Passive view.

In the case of *connect_to* message, the receiver has to send a change connection message to the indicated node so that it removes its connection with the connect_to sender. Only reachable nodes are in Active view.

Protocol Description

Connect Task:

```
1.  loop forever
2.      gap ← LOW − |neighbors|
3.      while gap > 0
4.          n ← random node
5.          send ⟨CONNECT, |neighbors|⟩ to n
6.          gap ← gap − 1
7.      sleep (connect_timeout)
```

from Active view

Protocol Description

Event Handlers-Procedures:

```
8.  upon receive ⟨CONNECT, d⟩ from n do
9.    if |neighbors| < HIGH then
10.     add_connection (n, d,true)
11.   else
12.     send ⟨REDIRECT,lowest degree neighbor⟩ to n

13. upon receive ⟨REDIRECT, n′⟩ from n do
14.   send ⟨CONNECT, |neighbors|⟩ to n′
```

Add n in Active view

```
15. upon receive ⟨CONNECT_OK, d⟩ from n do
16.   if |neighbors| < HIGH then
17.     add_connection (n, d,false)
18.   else
19.     send ⟨LEAVE⟩ to n

20. upon receive ⟨LEAVE⟩ from n do
21.   remove_connection(n)
```

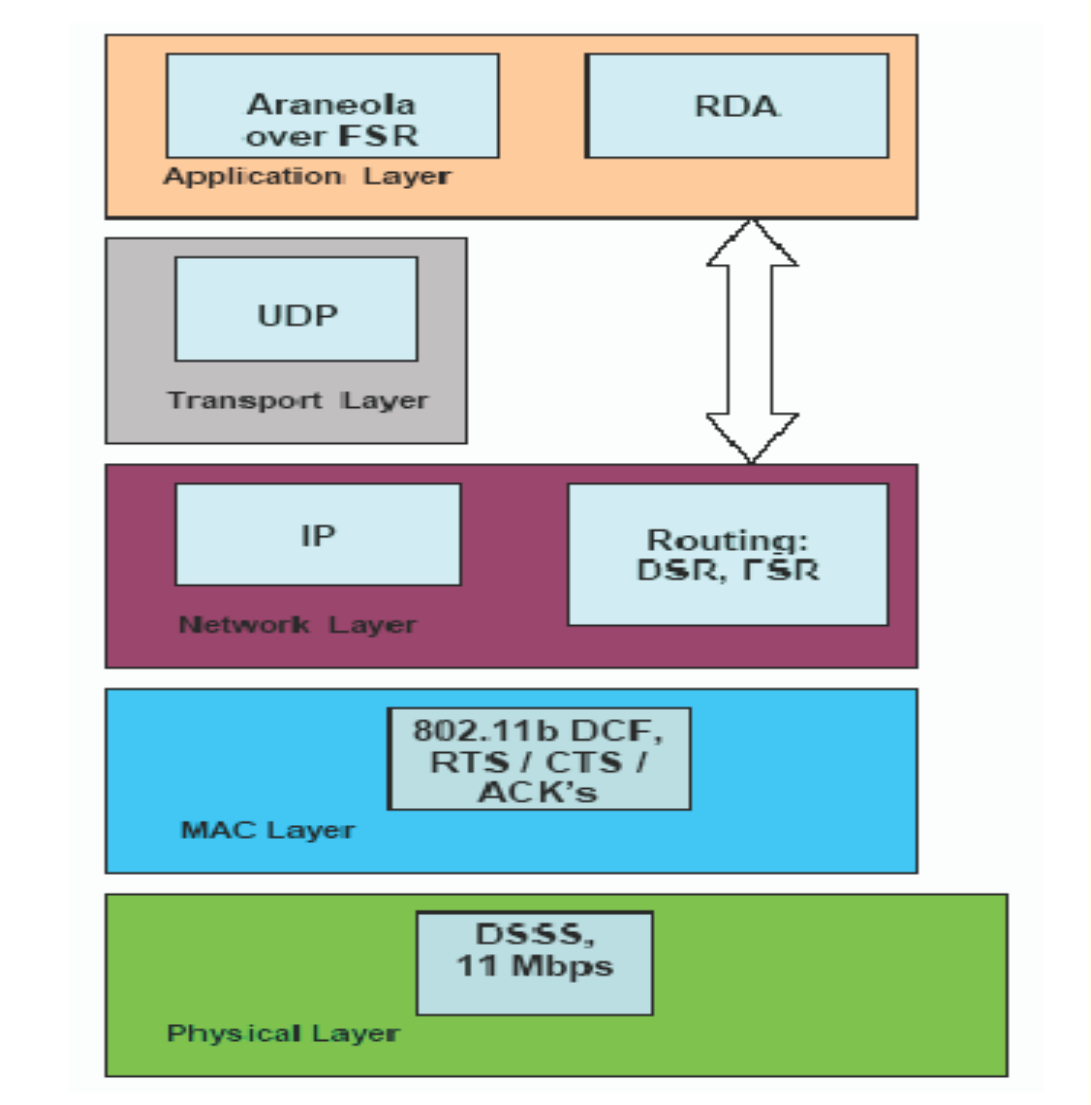
Only If n' is reachable, otherwise send(Connect) to reachable member of its choice

Route Driven Araneola

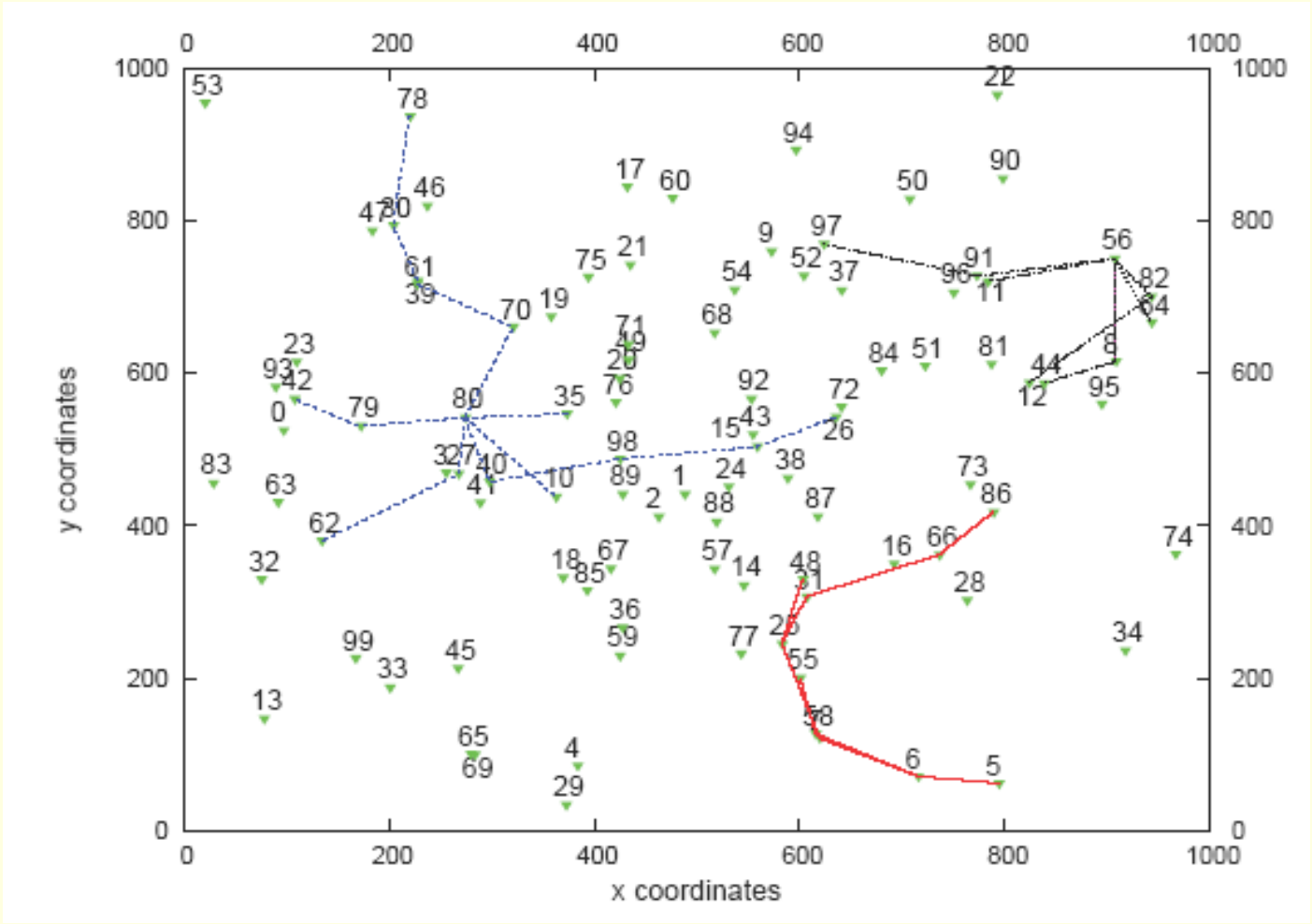
- Optimizations.
  - **Promiscuous** reception of routed packets.
  - **Mobility adaptation**. Parameter estimation (timeouts and high/low threshold) according to speed.
  - Topology awareness-TA
    - Weighted Active view according to hop distance to member. Nodes in proximity are selected with higher probability.
    - Affects randomness. However, mobility mitigates this problem.
  - Scoped flooding of DSR Join Requests. TTL = 3-5
    - Reduces Group Requests overhead
    - Provides a degree of topology awareness
    - Less prompt view population. Affects randomness

$$p_i = \frac{p_s^{h_i}}{\sum_{i\ in\ AV} p_s^{h_i} i}$$

Route Driven Araneola



RDA network snapshot



56->8, 11, 12, 44, 64, 91, 97
80->10, 26, 35, 42, 62, 78
5->6, 7, 48, 55, 86

Araneola over FSR

- The base Araneola protocol over FSR.
  - LPB-like membership management.
    - Non-route-driven.
  - Topology aware through FSR's topology table.
  - Promiscuous through IP's routing function.

Simulation Environment

- Glomosim. Library for discrete event parallel simulation
- RDA Glomosim implementation in the app layer.
  - Simulated protocol stack: UDP, IP, 802.11, Two-ray propagation/Ricean multipath fading model.

Experimentation Methodology

- Simulation Parameters:
  - 1000 m x 1000m area.100 Nodes. 600 sec. R=150m.
  - Mobility ranges from 0-20 m/s. For no mobility, grid placement. Otherwise, random waypoint model and random placement.
  - Message injection rate ranges from 0.05-2 msg/sec.
  - One multicast source.

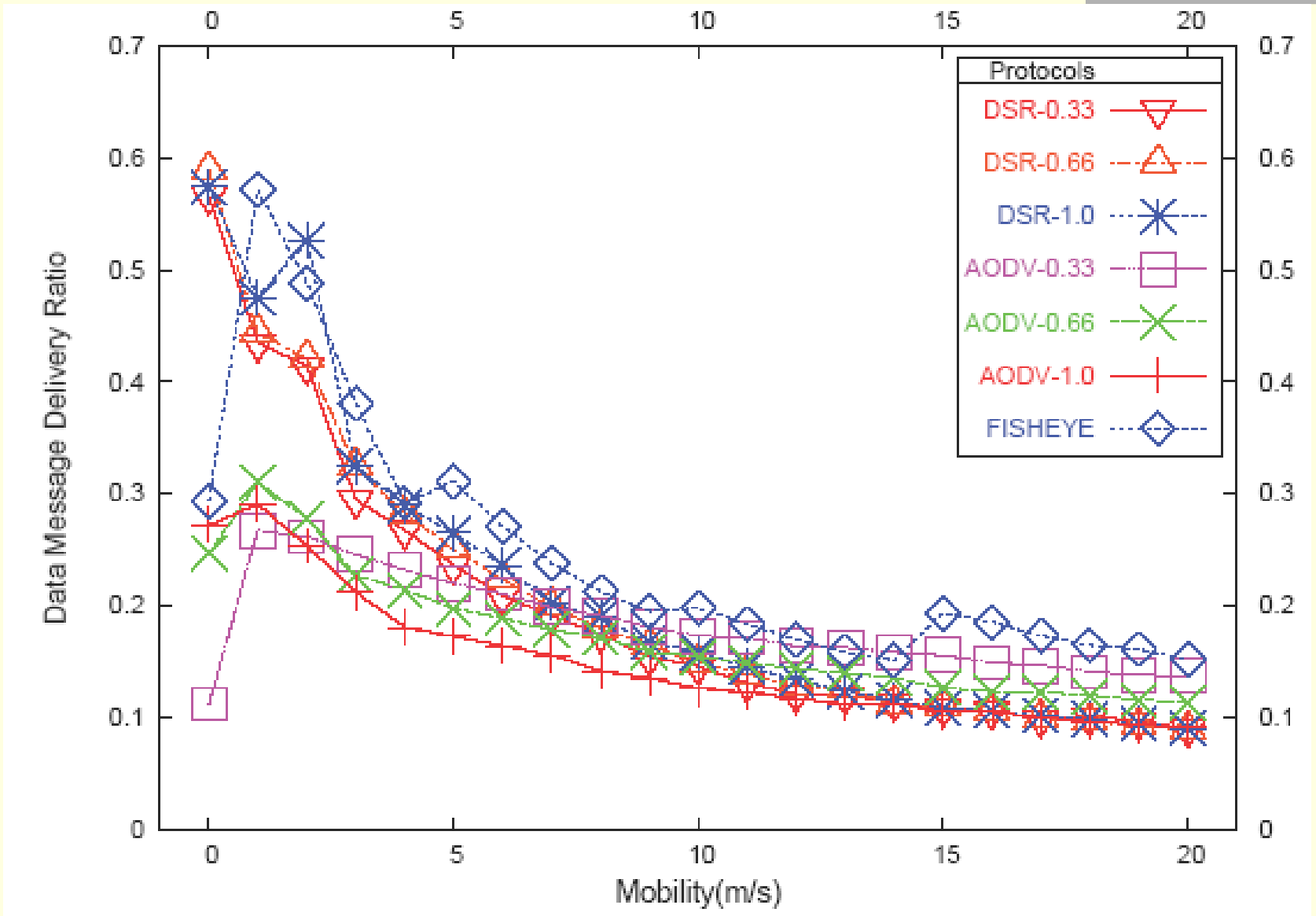
  - Performance metrics (MANET RFC 2501):
    - Reliability: data message delivery ratio
    - Control overhead: control bytes over delivered/transmitted data bytes.
    - Average end-to-end delay.

Experimentation Methodology

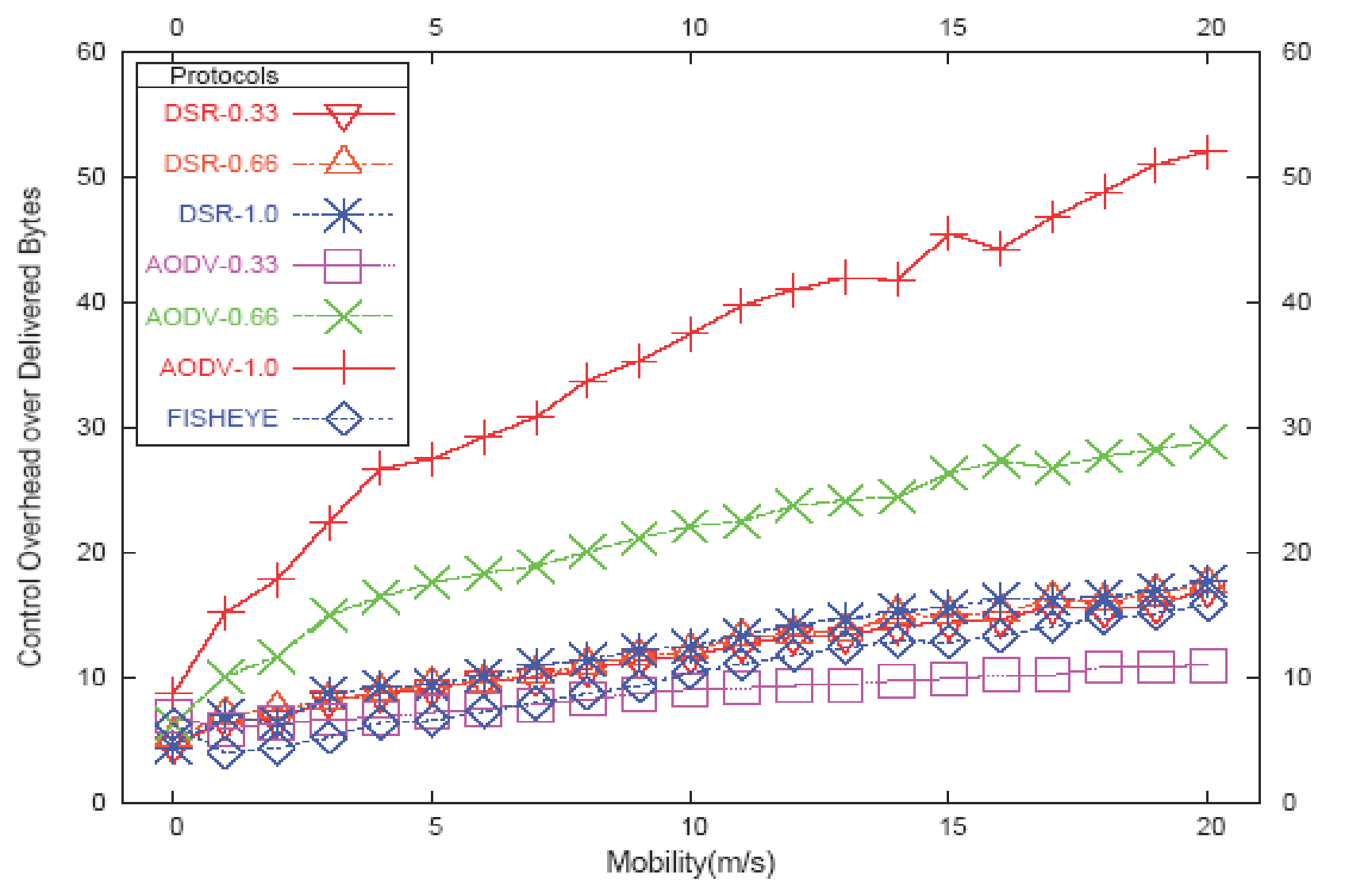
- Determining RDA overlay properties
  - Diameter determined by max overlay hop-count and end-to-end delay.
  - Connectivity approximated with minimum degree.

    (upper bound of connectivity)

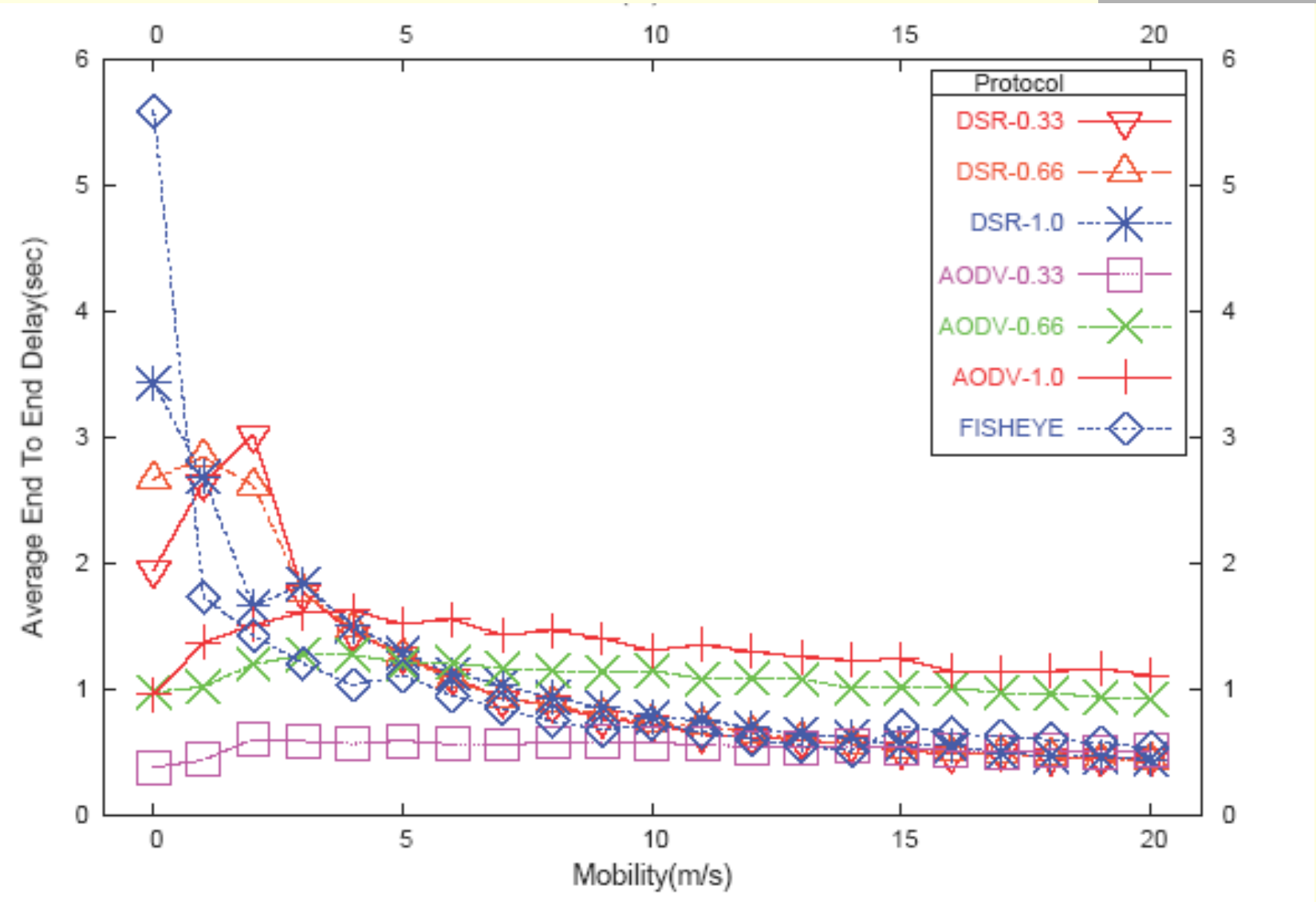
AODV/DSR/FISHEYE- Reliability



AODV/DSR/FISHEYE-Control Overhead



AODV/DSR/FISHEYE-End-to-End delay



Selecting Tunneling Strategy- Conclusions

- For a traffic load that is similar to the one imposed by ad-hoc Araneola DSR with relaying probability equal to 1.0 provides relatively reliable and scalable communication service.
- However, it is not superior to AODV and FSR for the whole spectrum of network configurations. It exhibits higher reliability than AODV, without entailing higher overhead, only for speeds below 8 m/s and packet injection rates lower than 0.5 pkt/s.
- It provides better reliability than FSR only under no mobility.
- For intense application layer multicast traffic, the majority of links is utilized and complete routing knowledge is required. In this case reactive protocols have high overhead and their reliability does not compare favorably to proactive protocols.

Comparing RDA variations

RDA variation	Characteristics
Baseline(RDA)	Low=5, High=10, GTO=5, CTO=15, DTO=20
RDA-MA	Mobility-adaptive. See tableVII.3
RDA-MAP	RDA-MA promiscuous. DSR passes routed Gossip and Data messages to RDA
RDA-MACR	RDA-MA that connects solely to reachable nodes
RDA-MATA	RDA-MA Topology aware. Weighted active view according to hop count
RDA-MASF	RDA-MA scoped flooding. Floods route queries with low TTL (5)
RDA-MADRP	RDA-MA repair data-pull. Performs repair pulls for missing messages
Araneola-MANRD	Araneola-MA over DSR that does not depend on route state. Non-route-driven. Promiscuous, DRP
Araneola-MALS	Araneola-MA over FSR (link-state) Uses RDA-MA parameters. Promiscuous, DRP, TA

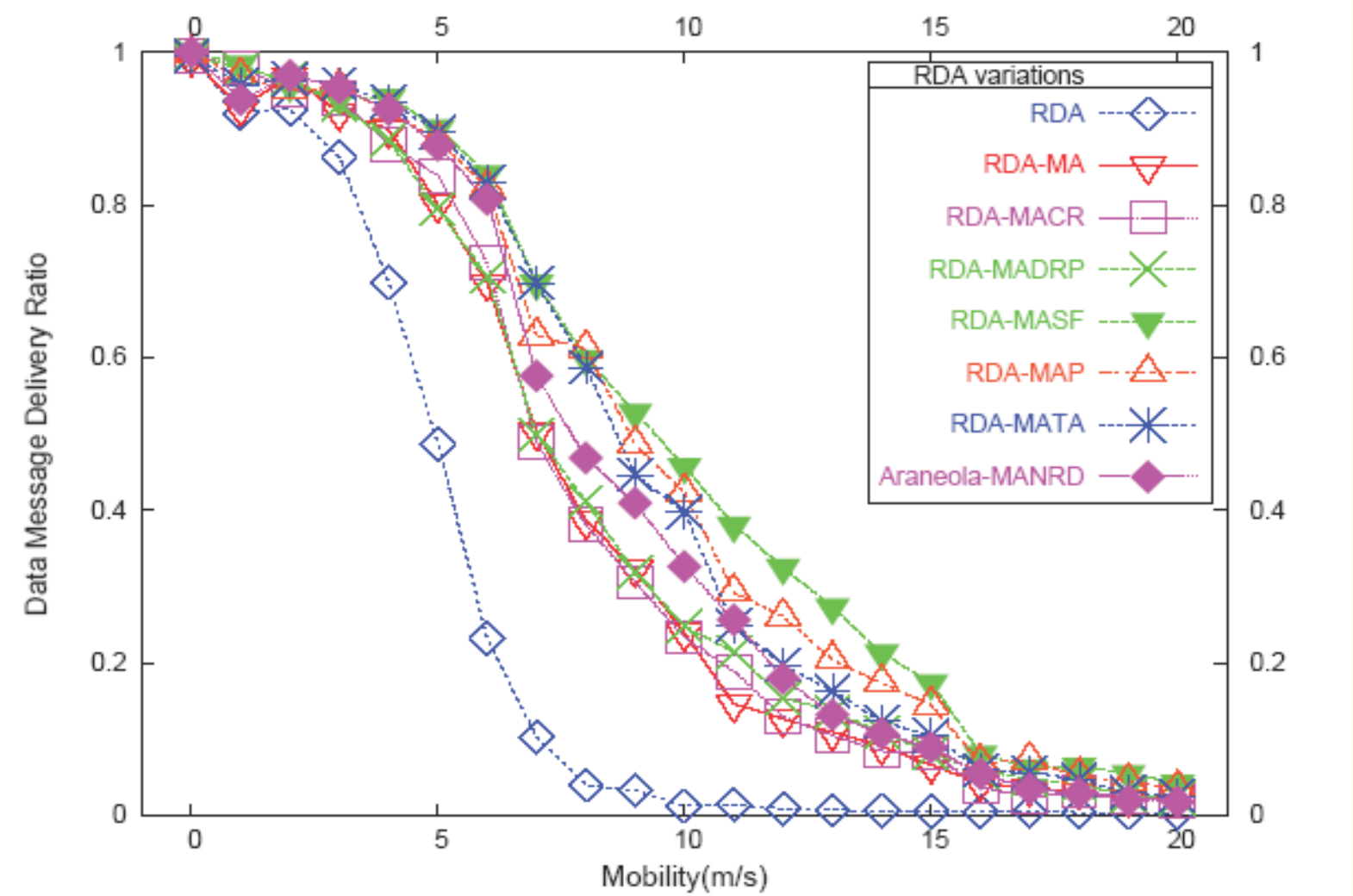
RDA-MA Parameters

RDA Parameters	0 m/s	1 m/s	2 m/s	3-4 m/s	5-6 m/s
Connect Timeout(sec)	20	15	10	8	5
Disconnect Timeout(sec)	30	20	14	12	8
Gossip Timeout(sec)	4	4	4	4	4
Failure Detection Timeout(sec)	12	12	12	12	12
Low Threshold	5	6	6	6	7
High Threshold	10	11	11	11	12
Active View Threshold	30	30	30	30	30
Passive View Threshold	30	30	30	30	30

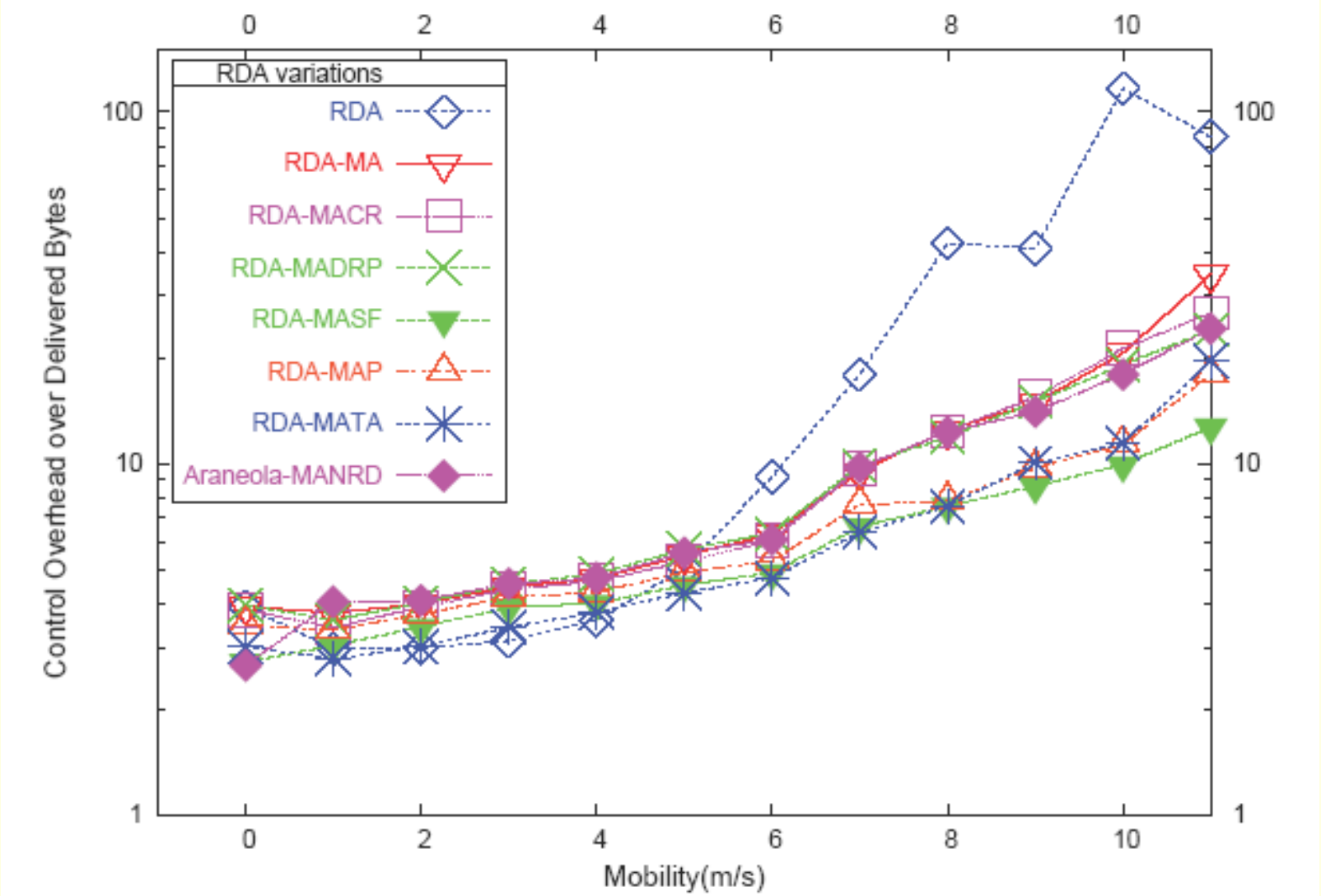
RDA Parameters	7-10 m/s	11-15 m/s	16-20 m/s
Connect Timeout (sec)	4	2	1.5
Disconnect Timeout(sec)	6	3	2
Gossip Timeout (sec)	4	4	4
Failure Detection Timeout(sec)	12	12	12
Low Threshold	8	9	10
High Threshold	13	14	15
Active View Threshold	30	30	30
Passive View Threshold	30	30	30

Table VII.3: Parameter values for Mobility-adaptive RDA

RDA-Reliability



RDA-Control Overhead



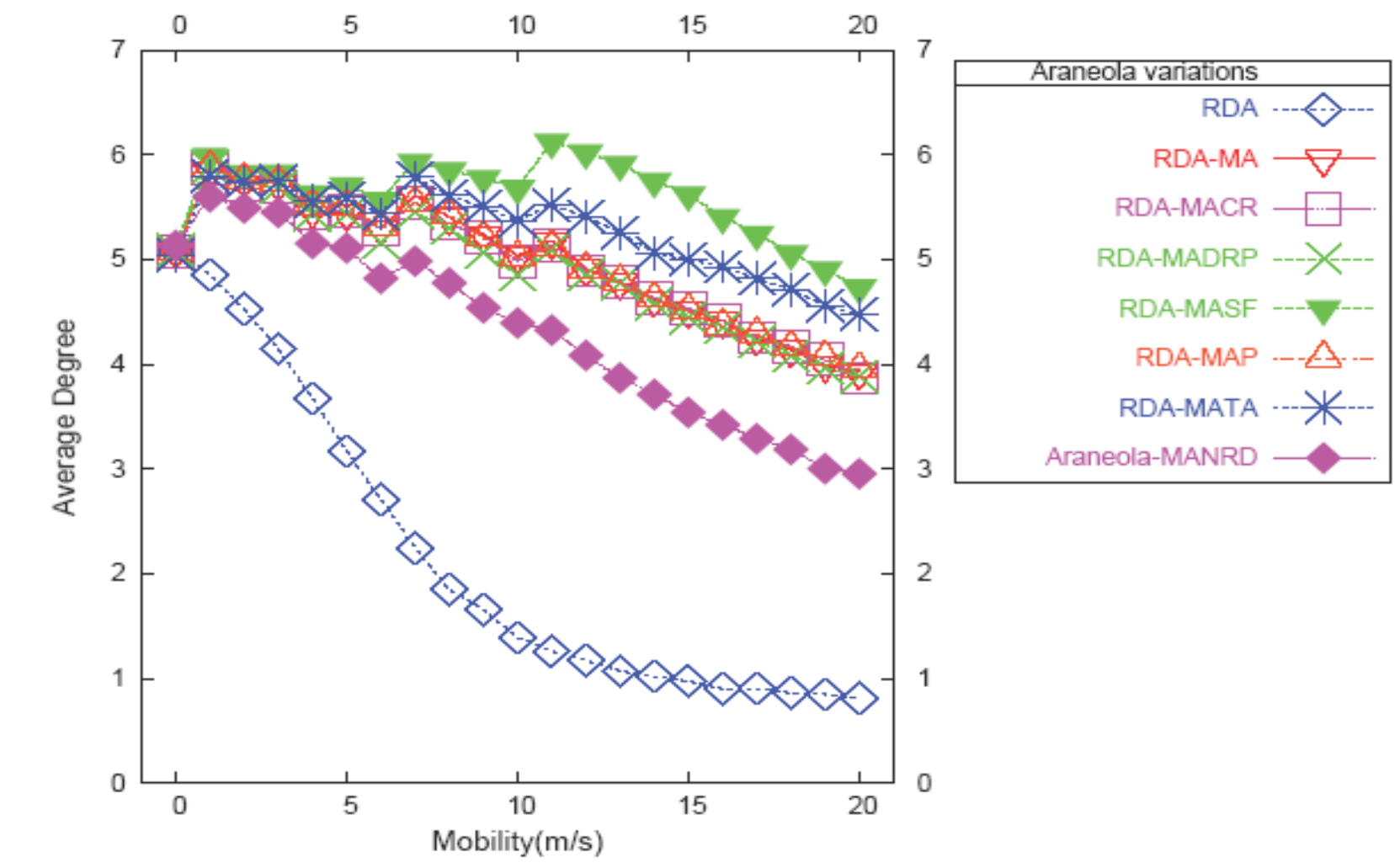
Evaluating RDA-Overlay Properties

- In random regular graph G(E,V) of degree k, diameter d(G) a.a.s satisfies:

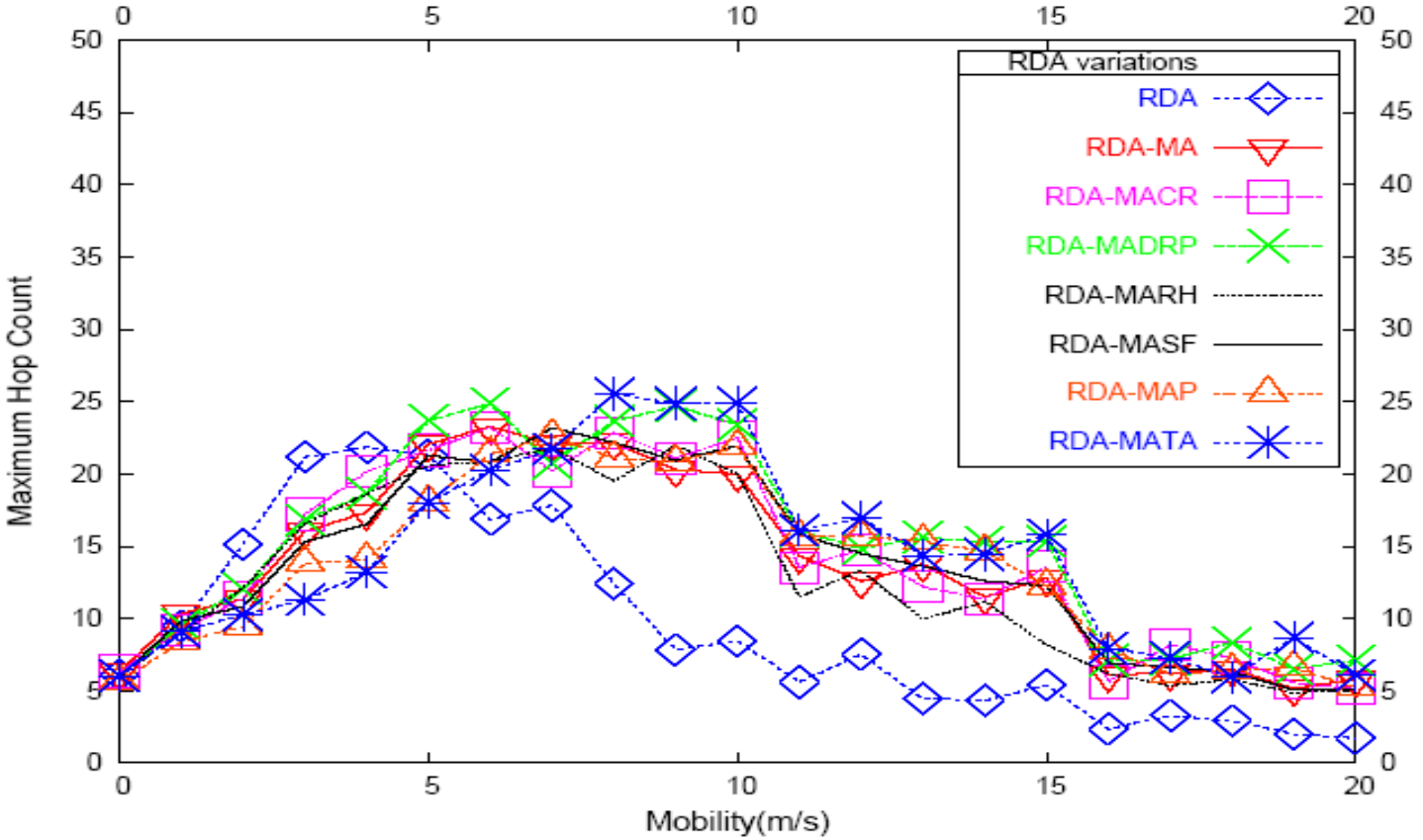
$$1 + \lfloor \log_{k-1} |V| \rfloor + \lfloor \log_{k-1} (\frac{k-2}{6k} \log |V|) \rfloor \leq d(G)$$
$$\leq \lceil \log_{k-1} ((2+\epsilon) k |V| \log |V|) \rceil$$

- For random regular G(E,V), V=100, d=4 => 4 < d(G) <9 (Not asymptotical, therefore not almost surely)

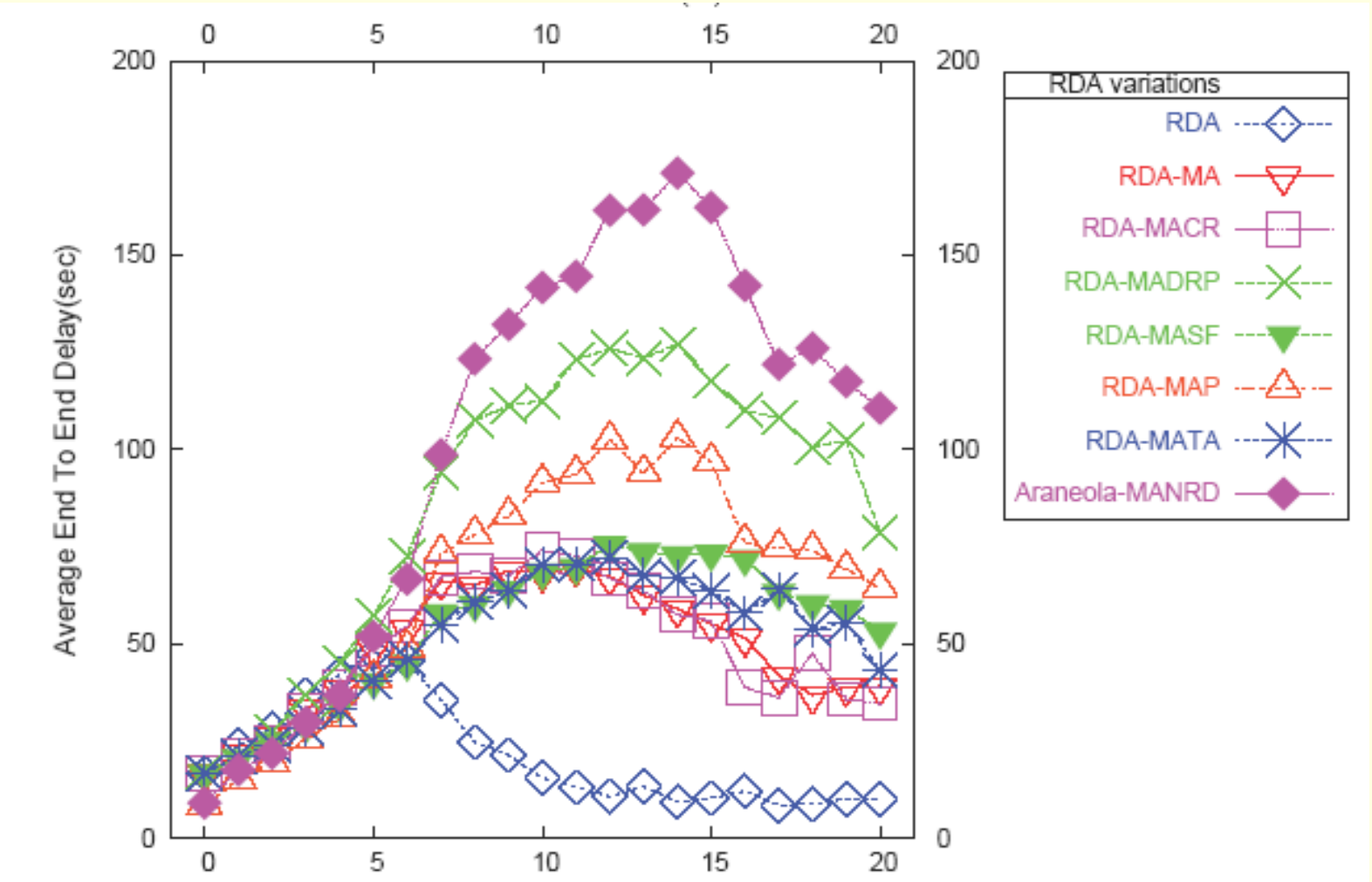
RDA-Overlay Properties- Avg Degree



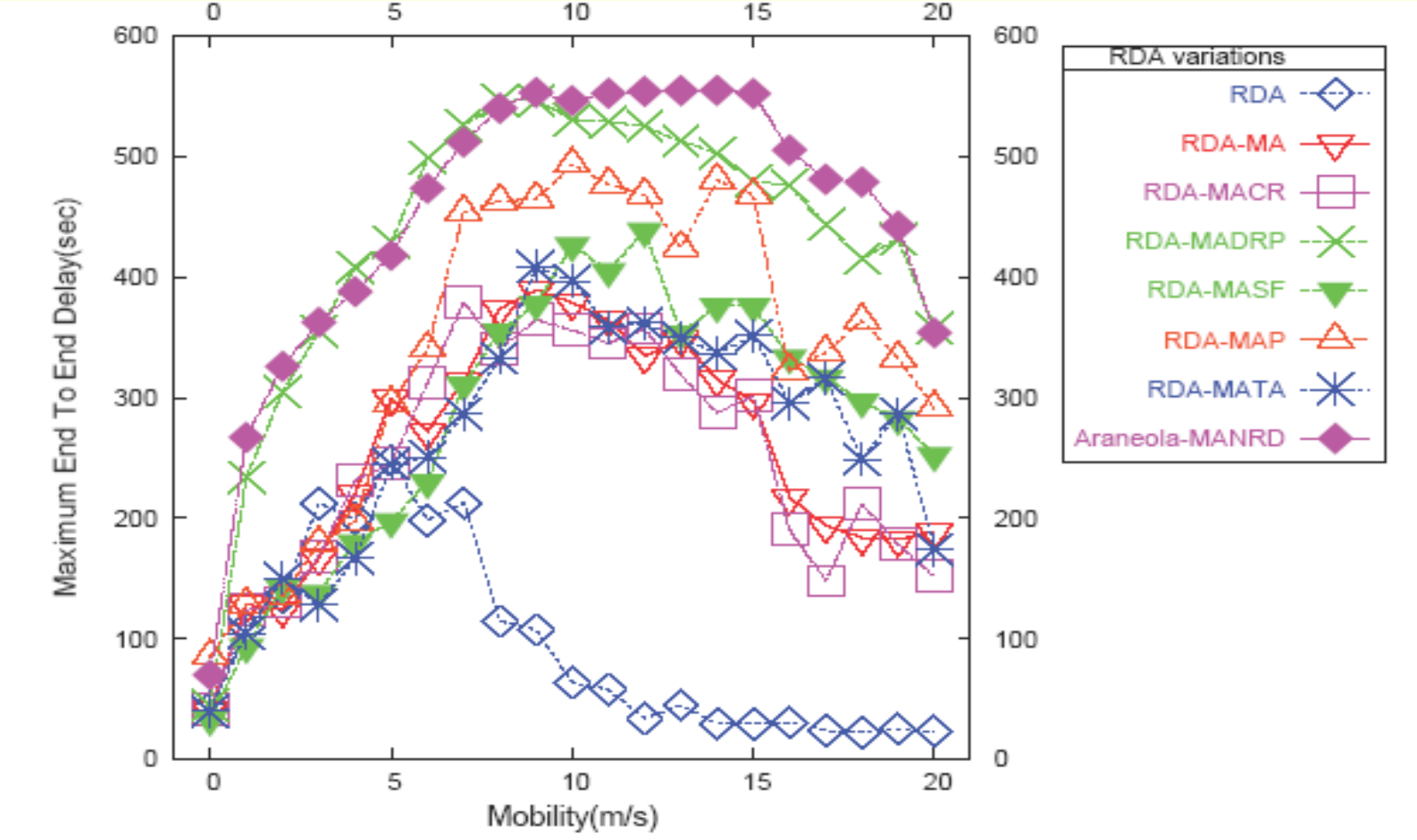
RDA-Max Hop Count



RDA-End-to-End Delay



RDA-End-to-End Delay



Overlay Properties-Conclusion

- Below 2 m/s the maximum hop count for all variations approximates the predicted diameter (between 5 and 10). Increases dramatically with increased mobility.
- For all variations and for 0-5 m/s the average hop count lies between three and seven.
- The maximum end-to-end delay is considerably above the theoretical predictions.
- Under low mobility, the inconsistency between the end-to-end delay results and the theoretical prediction indicates that, over time, a regular overlay is successfully created, however it is disrupted by temporary failures.

Overlay Properties-Conclusion

- End-to end-delay, and overlay degree indicate that, for low mobility (0-2 m/s), overlay creation is successful. For moderate mobility (4-10m/s) the overlay structure deteriorates.

- For non-mobility-adaptive RDA, under no and low mobility, we observe that the average degree approximates the low threshold (5). Thus, in a stationary or mildly volatile ad-hoc environment the overlay maintenance task is successful in creating the k-regular overlay.

Evaluating RDA-conclusions

- Adaptation to mobility is required for the protocol to cope with frequently changing topology.
- From the behavior of RDA-MAP and RDA-MATA, we determine that routing layer promiscuousness and topology awareness can improve all four examined aspects of performance.
- It is preferable not to remove entries that are associated with an non-connected node, since the overlay structure incurs temporary disconnections.
- Network proximity-based selection of neighbors (RDA-MATA, RDA-MASF) does not yield increased overlay diameter compared to RDA-MA, although randomness in overlay creation is reduced.

Evaluating RDA-conclusions

- The proposed application layer multicast overlay is a viable solution in a mildly volatile ad-hoc environment..

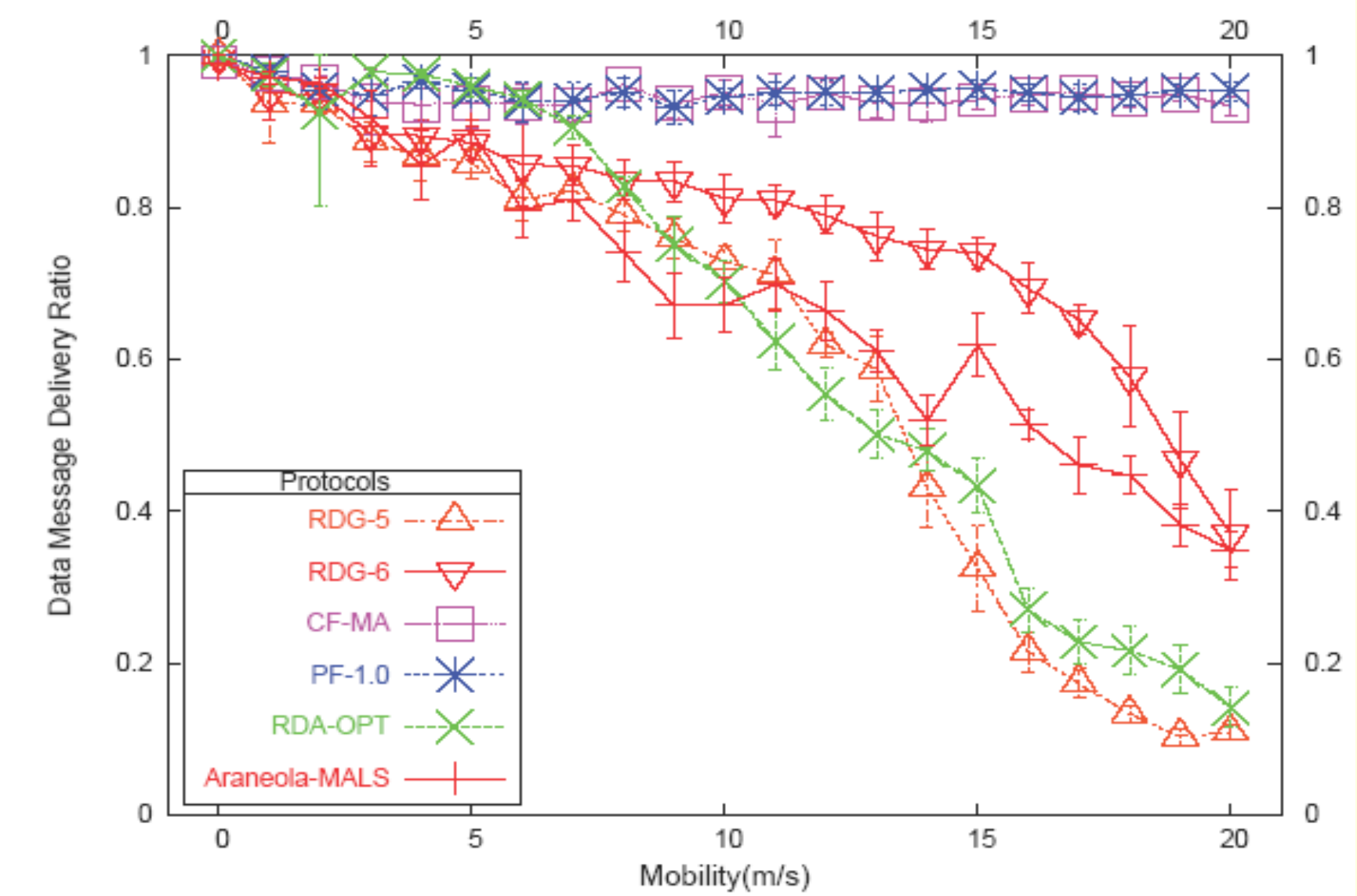
RDA vs Flooding and RDG

- Comparison among various multicast schemes:

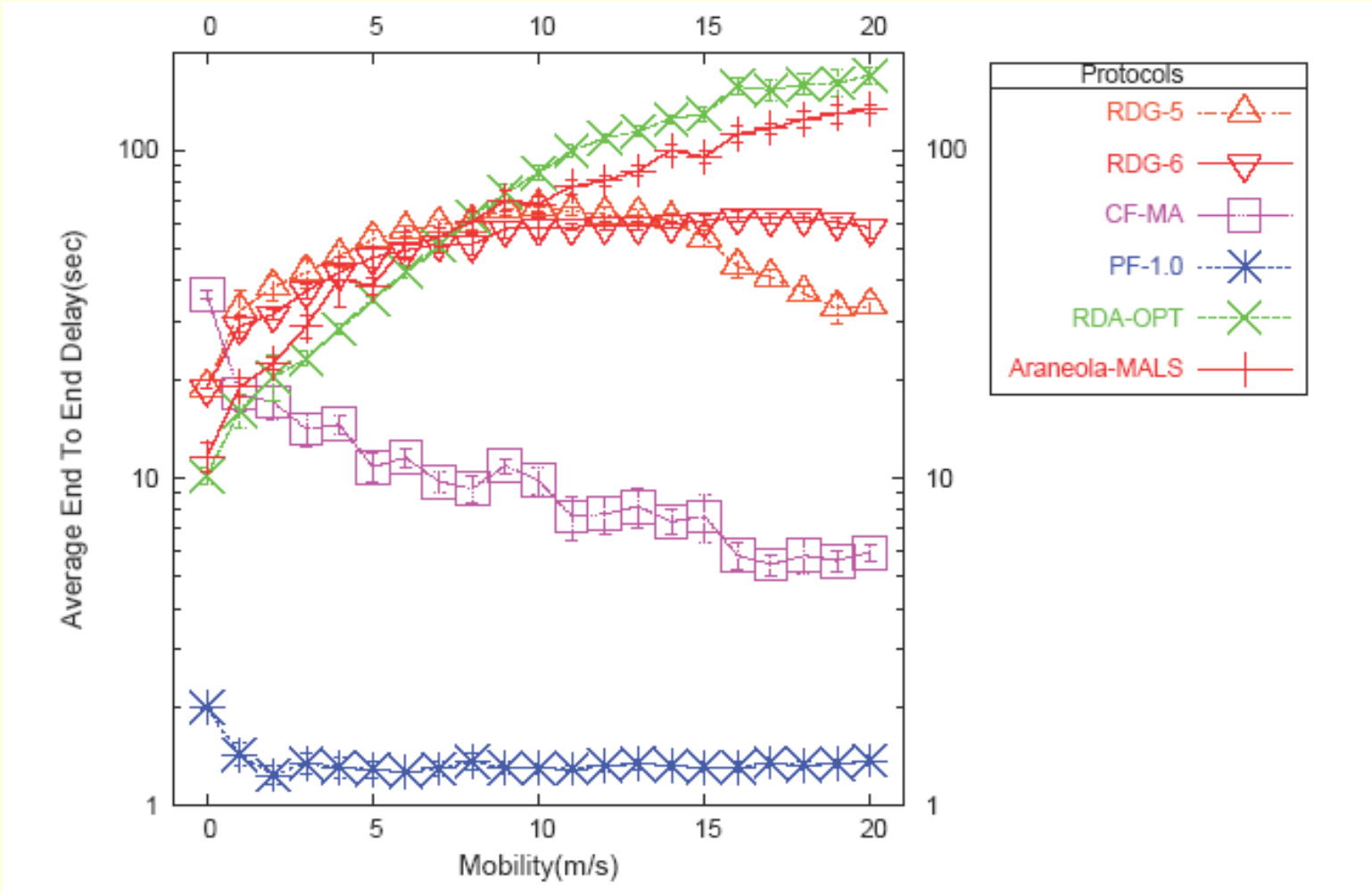
- Optimized RDA using all the beneficial modifications.
- RDG is a brute-force gossip protocol. The pure-gossip counterpart of RDA. Simulation evaluation of RDG with fan out 5 and 6. (Corresponds to RDA's target degree of 5).

  This is a gossip-pull variation of RDG.
- RDG is also topology aware and promiscuous.
- Flooding with relay probability 1.0
- Our simple variation of controlled flooding.

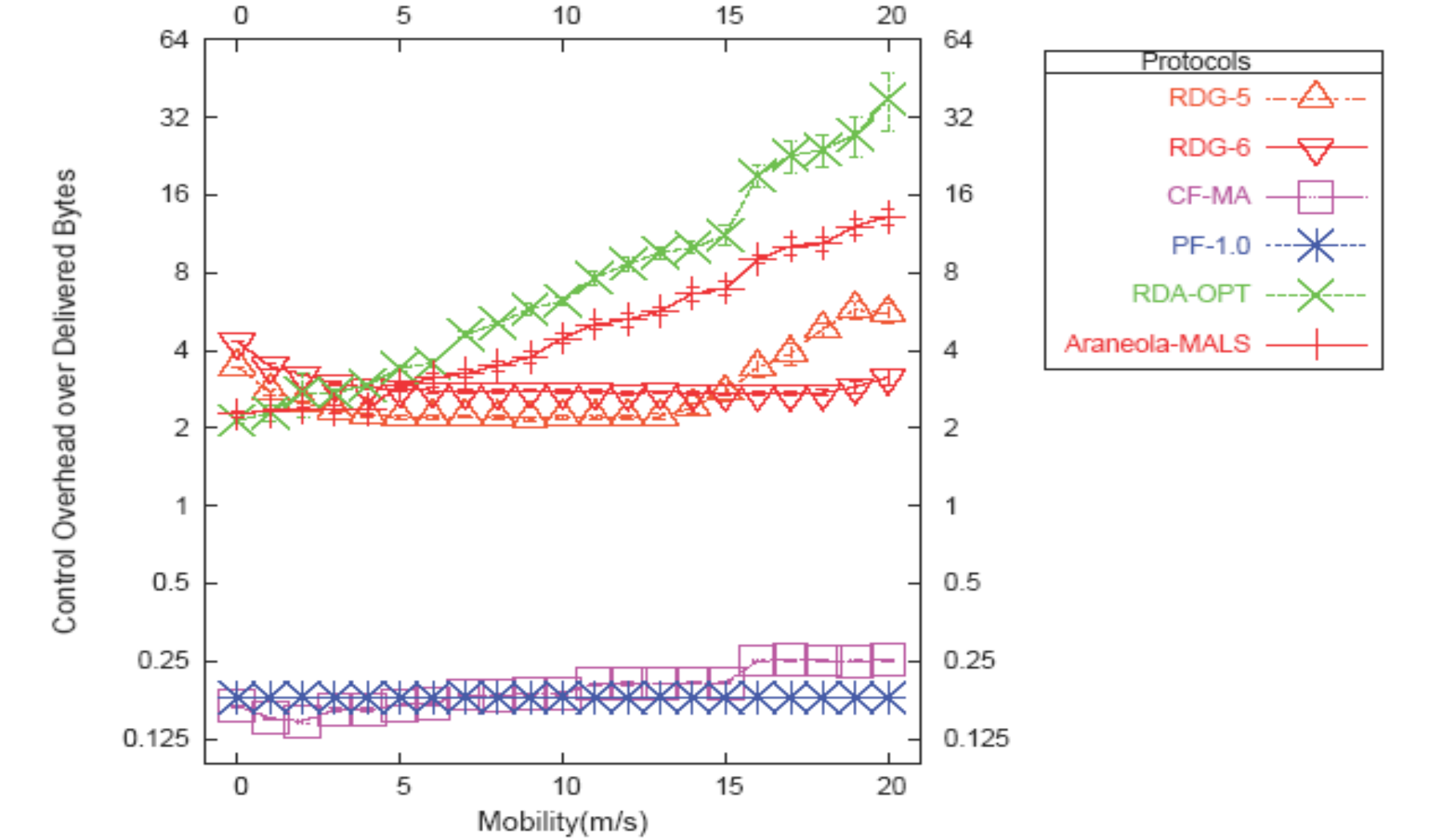
Protocol Comparison- Reliability



Protocol Comparison-End to End Delay



Protocol Comparison- Control Overhead



Protocol Comparison-Conclusions

- Flooding is the most reliable and efficient dissemination method, in a small network, in which all nodes are receivers.

  It is not substantially affected by mobility. Maintains high message delivery ratios, without imposing high control overhead.

- Optimized RDA yields higher reliability than both Araneola over link-state routing and RDG, for low mobility (0-7 m/s) and under low (0-2 msg/s) traffic load.

- For higher speeds RDG degrades more gracefully than the deterministic flooding protocols.
  - Due to the deterioration of the imposed regular graph structure and the increased signaling.

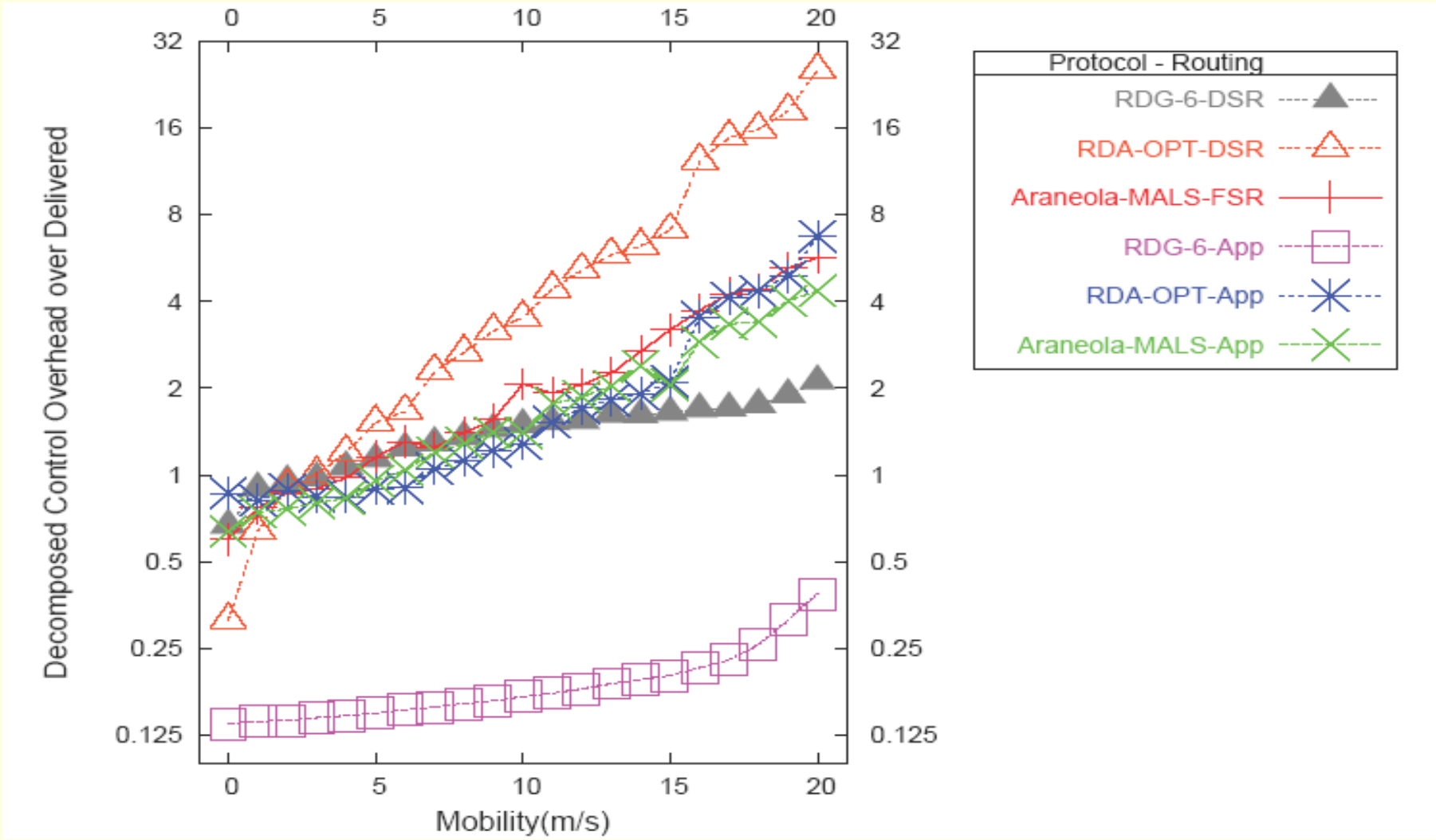
Protocol comparison-Conclusions

- For speeds below 5 m/s, RDA and Araneola-LS entail less signaling than RDG.
- For up to 10 m/s, RDA is more reliable than Araneola-LS, but this does not hold for higher speeds. The benefits of reactive routing protocols and of route-driven view cease to exist in a more dynamic topology and when the majority of the links is utilized.
- Under moderate and high mobility, reactive source routing consumes more network resources than ad-hoc link state routing.
- The investigated overlay message dissemination methods are not suitable for real-time communications. They exhibit one to two orders of magnitude higher latency and jittering than other multicast mechanisms.

Decomposition of control overhead

- Decomposition of overhead in routing and application layer signaling.
- The first includes DSR route requests/replies, FSR update messages, source routing header in the IP options field, UDP/IP/802.11 headers and 802.11 control packets transmitted for DSR or FSR frames.
- The latter includes application layer protocol headers, overlay maintenance headers and 802.11 control packets transmitted for Araneola frames.
- Control bytes are distinguished at the 802.11 module of the simulation.

Decomposition of control overhead



Decomposition of control overhead

- Since RDG does not transmit application layer control packets, routing overhead is always larger than application overhead. This ratio increases as route errors become more frequent with increased speed.

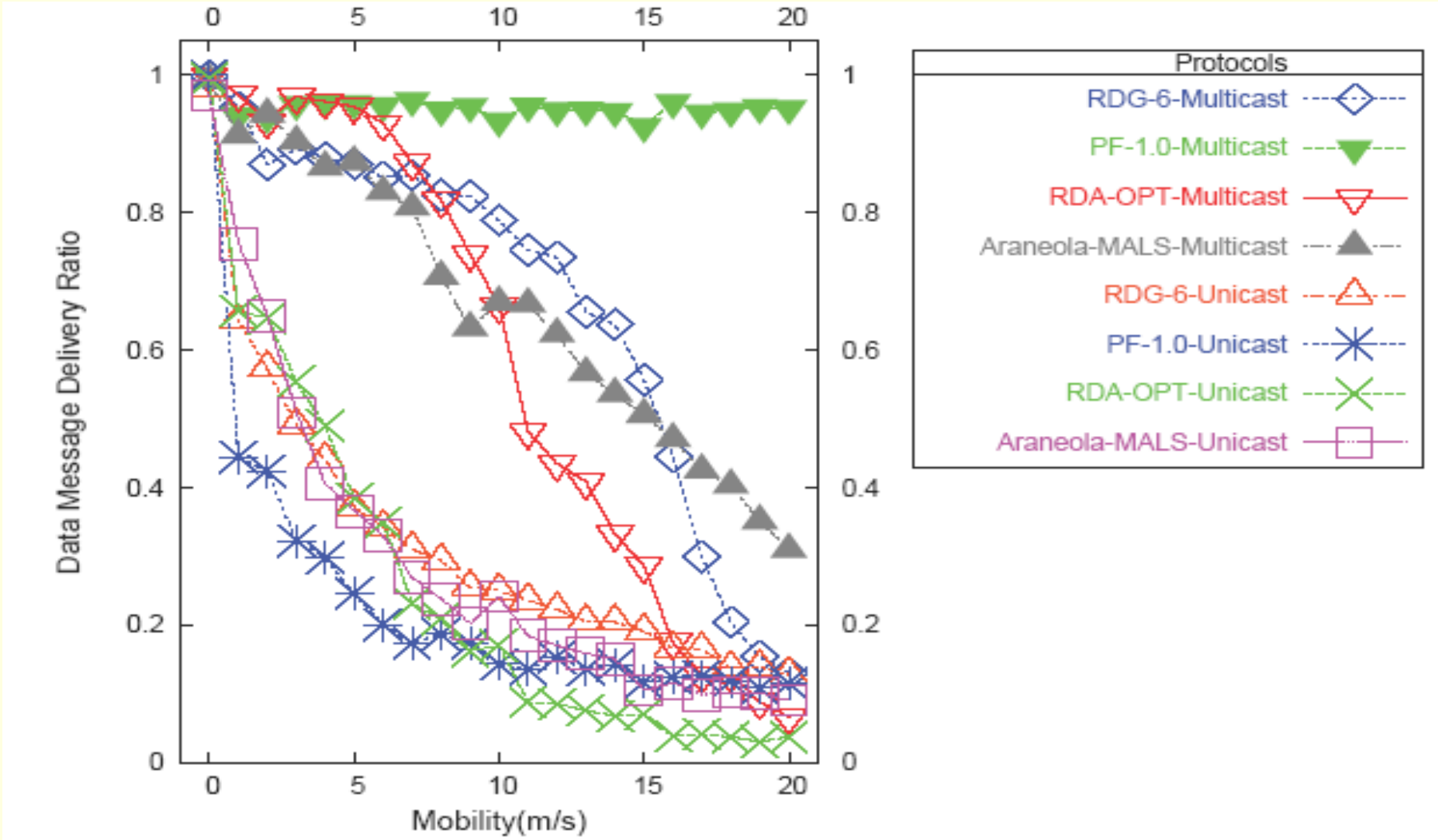
- For RDA at low speeds, the dominant component is application signaling but the ratio over routing signaling decreases with speed. As DSR attempts to tackle frequently changing connectivity, which invalidates cached routes, it initiates more frequent route discoveries, inducing contention. The routing overhead is not inhibiting RDA multicast for low mobility.

- The routing layer signaling of Araneola-LS is less than RDA-OPT's, except of the case of very low mobility. In Araneola-LS the routing to app layer signaling ratio does not increase. Thus, ad-hoc link-state routing is less costly than reactive source routing for the Araneola's mode of communication, under moderate and high mobility.

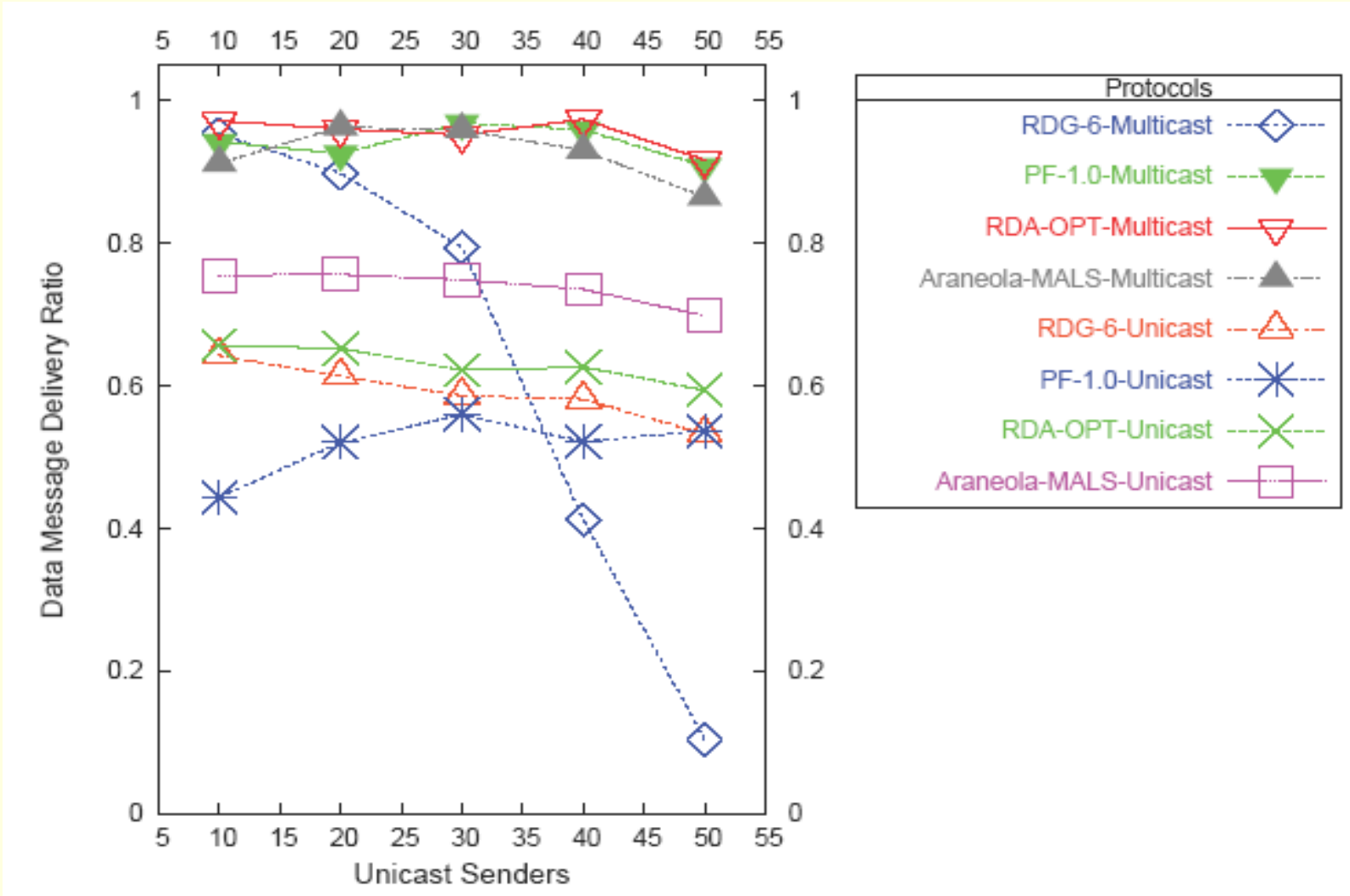
RDA vs Flooding and RDG under unicast traffic

- Both multicast and unicast load is injected in the network.
- Number of unicast senders ranges from 10-50.
- Unicast senders and destinations randomly selected among the nodes. Random movement too.
- Seeking to determine whether the unicast traffic amortizes the cost of application layer multicast.
- Expect overlay scheme to compare more favorably to flooding than in the absence of unicast traffic.

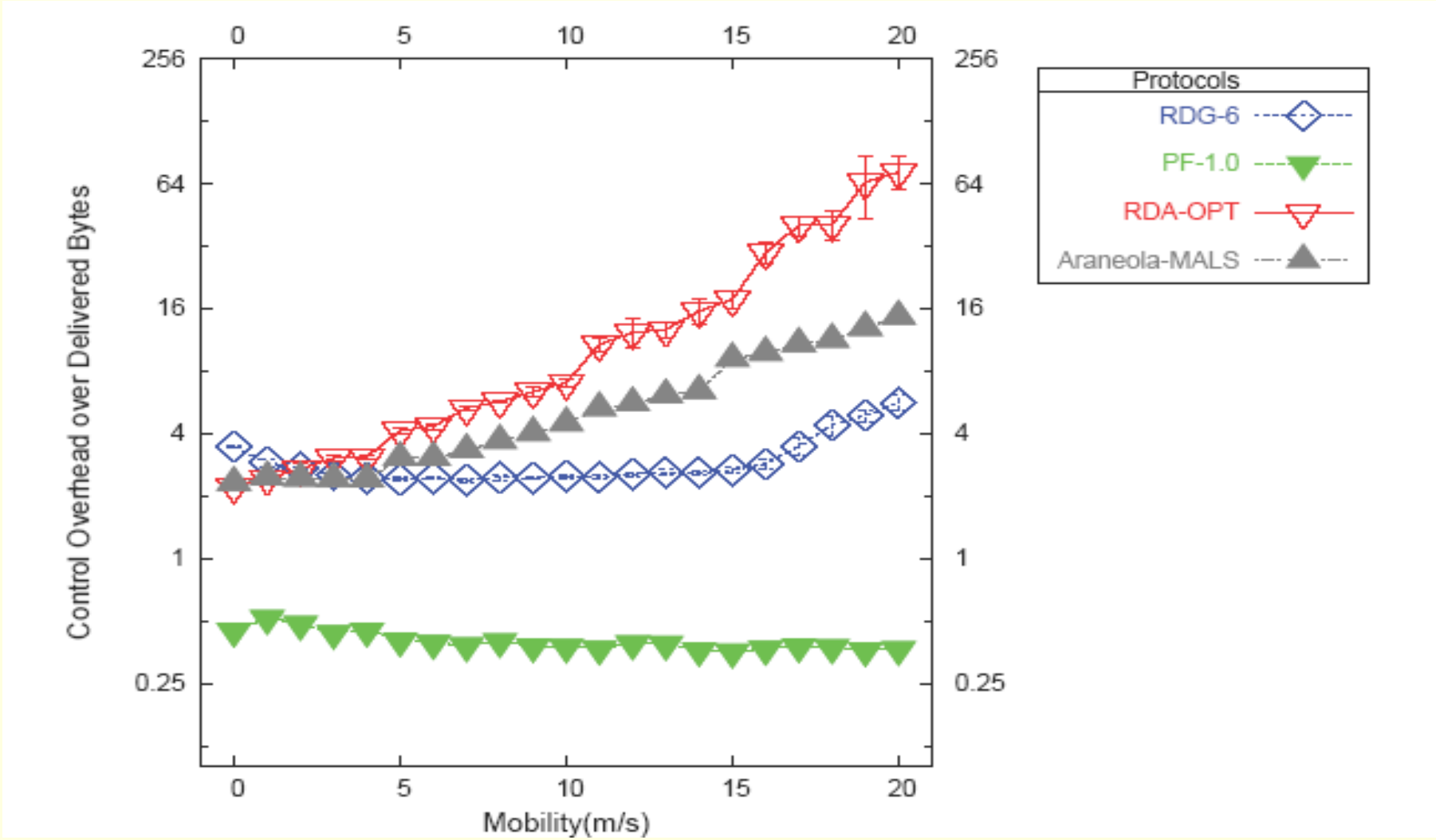
Protocol comparison under unicast traffic-Reliability



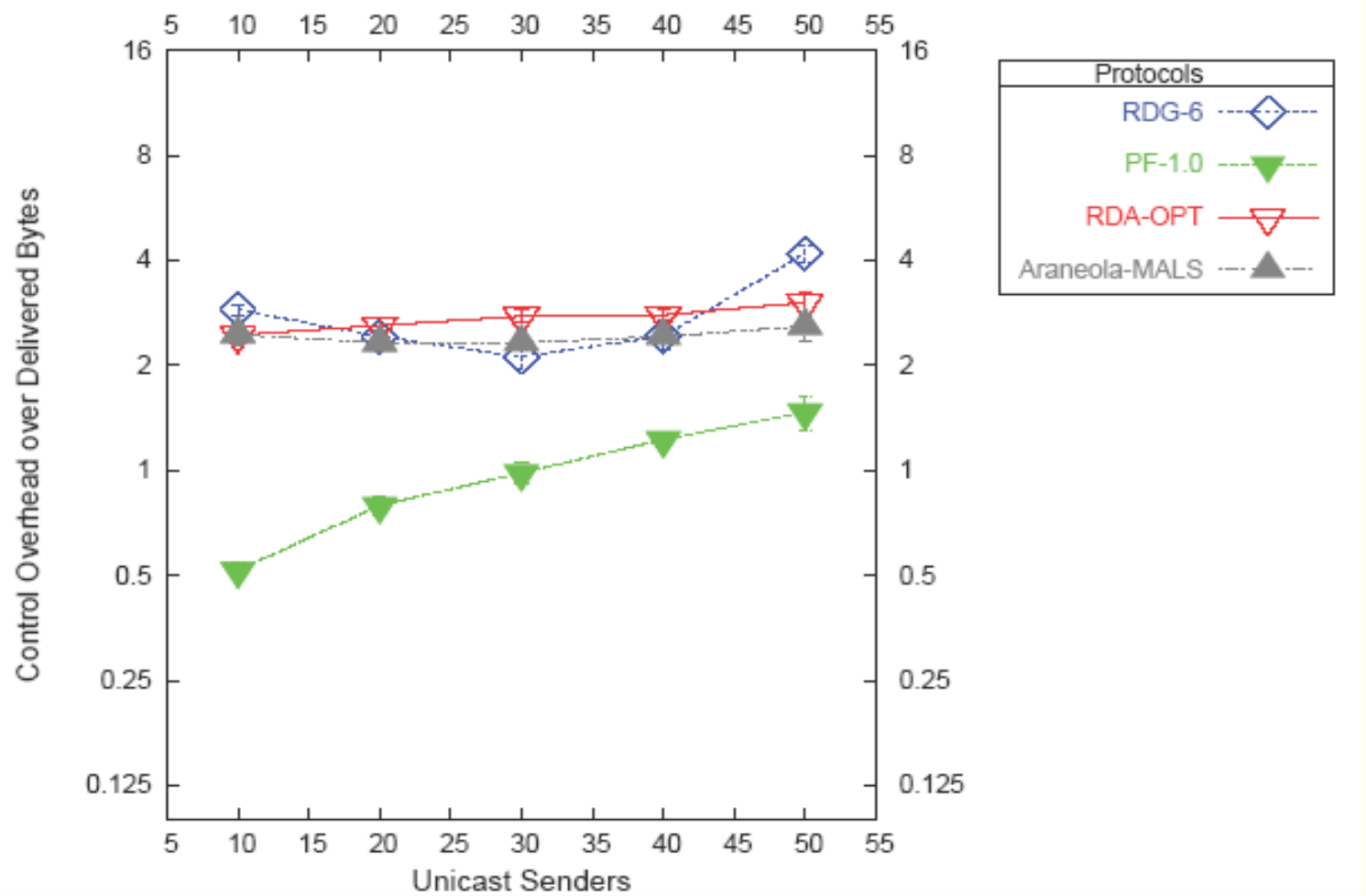
Protocol comparison under unicast traffic-Reliability



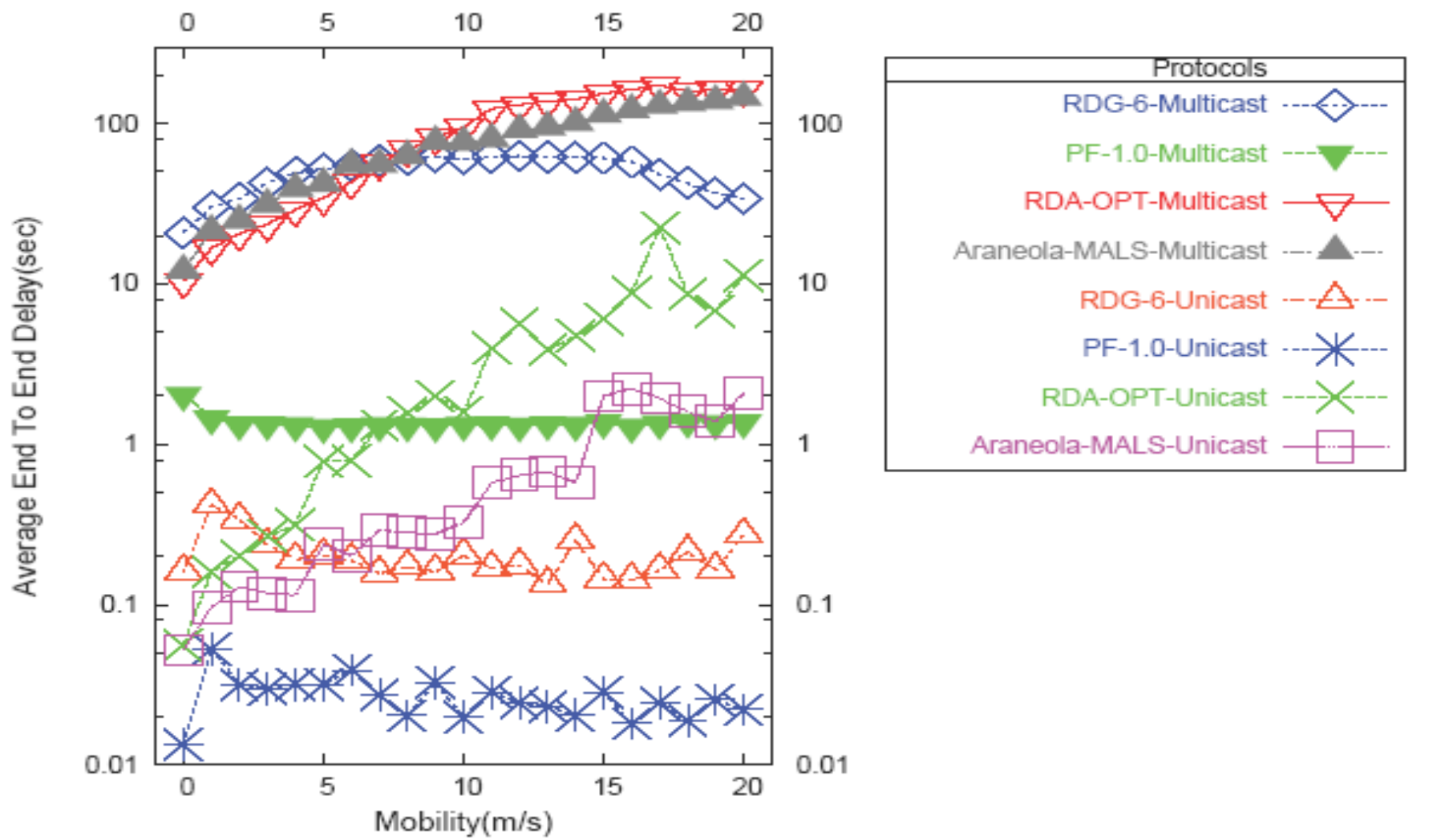
Protocol comparison under unicast traffic-Control Overhead



Protocol comparison under unicast traffic-Control Overhead



Protocol comparison under unicast traffic-End-to-End Delay



Protocol comparison under unicast traffic-Conclusions

- RDA-OPT outperforms RDG-6, in terms of multicast reliability, for the same range of mobility as for multicast-only traffic.
- Similar observations for Araneola-MALS.
- Flooding is still more reliable than RDG, RDA and Araneola-MALS, especially for speeds over 6 m/s.
- Unicast traffic contributes in updating the routing tables and amortizes the overhead that the routing protocols introduce.
- Ad-hoc Araneola, utilizes the underlying tunneling mechanism for both message dissemination and overlay maintenance, and is more benefited by background unicast traffic. It incurs less overhead than RDG as the unicast senders increase and exhibits much higher multicast reliability.
- The application layer schemes provide higher than flooding unicast reliability.

Discussion and Conclusion

- We address the problem of reliable and scalable ad-hoc multicast, focusing in overlay-based epidemic message dissemination algorithms.

- Previous work has determined that in an environment of complete connectivity and ignoring underlying topology, flooding over an expander connection graph is more reliable and requires less message overhead than gossip.

Discussion and Conclusion

- Ad-hoc Araneola performance is satisfactory. Deterministic flooding suitable for the ad hoc environment too. Given:
  - low mobility (below ~ 10m/s)
  - fine-tuning of routing mechanism
  - topology awareness
  - mobility adaptation,
  - route-driven membership
  - routing layer promiscuous operation
- The routing signaling is used toward building a structure for effectively controlling the message dissemination process.
- Cross-layer design is a necessity.

Discussion and Conclusion

- Per-hop Araneola attempts to reduce control overhead, directly interfacing with the link layer.
  - The additional broadcast control traffic load, saturates PHA network, inhibiting neighbor discovery and overlay maintenance tasks.
- The routing protocol provides additional overlay connectivity, increasing randomness and enhancing overlay properties.
  - RDA yields higher control overhead, yet reliability is significantly improved.
- Meta-data based controlled flooding and plain flooding are the most efficient multicast methods.
  - They incur signaling overhead solely by meta-data gossip and duplicate packets respectively.

Discussion and Conclusion

- Unicast traffic provides route resolution which is used by the multicast functions. Similarly, unicast traffic utilizes paths resolved for multicast leveraging the gains from the existence of route state in the network.

- Gossip exhibits substantially inferior to ad-hoc Araneola performance, under high unicast traffic load.

- Flooding demonstrated low unicast reliability in these environments. It does not take advantage of the existing routing infrastructure and interferes with flood-based route discoveries, causing unicast traffic packet losses.

Future Directions

- Under moderate and high mobility, the overlay structure deteriorates significantly. The routing mechanism fails to deliver packets reliably. We address it by implementing further optimizations to make the overlay more resilient in dynamic topologies.
  - Mesh-based RDA. Hear promiscuously all gossip and data messages regardless of MAC destination.
  - Threshold adaptation according to observed local connectivity.
- Evaluations under Reference Point Group Mobility model and node cluster models.
- Experiments in large scale exploiting the parallel computation capability of Glomosim.
  - Study scalability issues of routing protocols
  - Better experimental validation of random regular graph analysis results (a.a.s.)

Future Directions

- Derive analytical model using random regular graphs based on previous work that uses random graphs.
  - Prob. Of Node Disconnectivity = f(degree, network size)

Thank you. Questions?

References

SCAMP Ayavaldi Ganesh, Anne-Marie Kermarrec, Laurent Massulie.
{\it Peer-to-Peer lightweight membership service for large scale group communication}.
Proceedings of the Third International COST264 Workshop (NGC 2001).

LPB P. Th. Eugster, R. Guerraoui, S. B. Handurukande and P. Kouznetsov.
{\it Lightweight Probabilistic Broadcast}.
In Proceedings of the International Conference on Dependable Systems and Networks (DSN 2001).

Araneola}Roie Melamed and Idit Keidar.
{\it Araneola: A Reliable and Scalable Multicast Service for Dynamic Environments}.
In Proceedings of the 3rd IEEE International Symposium on Network Computing and Applications
(IEEE NCA), August-September 2004.

GossipVsDeterministic}M.J Lin, K. Marzullo, and S. Masini.
{\it Gossip vs Deterministic Flooding: Low message Overhead and High Reliability for
Broadcasting in Small Networks}.
In 14th International Conference on Distributed Computing (DISC 2000), Toledo,
Spain, 4-6 Oct. 2000), pp. 253-267.

DistributedConstructionExpanderGraphs}Ching Law and Kai-Yeung Siu.
{\it Distributed Construction of Random Expander Graphs}
Proceedings of Infocom 2003.

RDG Jun Luo Patrick, Th. Eugster and Jean-Pierre H.
{\it Route Driven Gossip: Probabilistic Reliable Multicast In Ad-hoc Networks}.
In Proceedings of INFOCOM 2003.

AODV}C. E. Perkins and E. M. Royer.
{\it Ad-hoc On-Demand Distance Vector Routing}.
Second IEEE Workshop on Mobile Computing Systems and Applications, pp.90-100, February 1999.

Diameter}B. Bollobas and W. F. de la Vega.
{\it The Diameter of Random Regular Graphs}.
Combinatorica, 2, (1982), 125-134.

NumberOfNeighborsConnectivity}Feng Xue, P. R Kumar.
{\it The number of neighbors needed for connectivity of wireless networks}.
Wireless Networks. March 2004, Volume 10, Issue 2.

MANETRFC}S. Corson.
{\it Routing Protocol Performance Issues and Evaluation Considerations}.
IETF MANET RFC 2501.
\PointGroupMobility} X. Hong and M. Gerla and G. Pei and C. Chiang.
{\it A Group Mobility Model for Ad-hoc Wireless Networks }.
ACM/IEEE MSWim, pages 53-60. 1999.

\bibitem{ProbabilisticReliableDissemination}Anne-Marie Kermarrec, Laurent Ma
Ayalvadi J. Ganesh.
{\it Probabilistic Reliable Dissemination in Large-Scale System}.
IEEE Transactions on PADS, pp 248-258.