

Overlay Multicast In Mobile Ad-Hoc Networks Using Araneola

Michael Sirivianos

University of California, Irvine

Abstract

Araneola, a system for scalable reliable multicast in dynamic wired network environments, exploits the good properties of random regular graphs (high connectivity and diameter that grows logarithmically with network size), to provide reliable, scalable and efficient message dissemination. We investigate the suitability of Araneola's probabilistic overlay creation and deterministic data dissemination mechanism in the wireless ad-hoc environment.

Araneola relies on a membership view tracking mechanism. Existing partial view membership protocols for the wired networks do not adapt to the limited ad-hoc connectivity and do not efficiently utilize the resource constraint medium. Hence, we replace the LPB-like membership track algorithm with the route-driven view mechanism of a rudimentary gossip protocol (RDG), which associates a view with the route state of a reactive routing protocol. In addition, we build an Araneola-based protocol that relies on ad-hoc link-state routing and LPB-like view acquisition. We perform numerous modifications on the base Araneola to allow it to cope with the idiosyncrasies of the mobile wireless environment.

We perform extensive simulation-based evaluation of our approach in environments with both multicast and unicast traffic. We compare it to application layer gossip and to link-level flooding. The experimental results enable us to determine the range of ad-hoc network configurations in which deterministic overlay flooding is a plausible application layer multicast technique.

Key words: Ad-hoc Multicast, Regular Graphs, Unicast, Gossip, Deterministic Flooding, Application Layer Multicast (ALM).

Email address: msirivia(at)uci.edu (Michael Sirivianos).

1 Introduction

A major challenge in mobile ad-hoc environments is a reliable, scalable and efficient multicast service. A class of proposed ad-hoc multicast systems performs in-network multicast with the goal of efficiently utilizing network resources. On-demand protocols proposed in [?] and [?] have the advantage of promptly adapting to topology and membership changes. In addition, they maintain the minimum possible state for as long as it is needed. These protocols either build and maintain a tree structure ([?]) or employ mesh networking ([?]). Work in [?] and [?] improves on [?] by introducing a congestion controlled phase error repair phase. [?] presents a proactive protocol that performs multicast using a shared tree link-level structure.

The protocols presented in [?,?,?,?] perform application layer multicast (ALM), using unicast tunnels to create virtual meshes or shared trees to connect group members. Topology volatility due to mobility and failures are handled solely by the underlying unicast protocols.

Comparative evaluations [?] have shown that the above multicast schemes demonstrate performance limitations when tested under the adverse network conditions of the ad-hoc environment. Tree-based protocols have limited adaptability and high reconfiguration control overhead, while mesh-based protocols still do not succeed in providing a highly reliable service.

The above suggest the need for an adaptable multicast service that considers the lack of determinism in terms of topology and group membership, as well as the resource scarcity in mobile multi-hop networks. The non-deterministic nature of ad-hoc networks motivates the use of epidemiological (probabilistic) protocols. These protocols make limited assumptions about the underlying network and rely on epidemic dynamics to provide, scalable and reliable message dissemination. In [?], gossip is used to improve the reliability of a reactive in-network multicast protocol. Requests for missing messages are issued to randomly selected nodes in the multicast group, to which routes are resolved in the multicast tree creation process. Route Driven Gossip (RDG) [?] is a gossip-based multicast protocol that relies on a unicast abstraction and employs an innovative membership view mechanism.

However, gossip-based protocols, incur increased control overhead without providing reliability and end-to-end delay guarantees. Their connection graph can be modeled as a random graph [?], which requires high fanout in order to ensure that the graph remains connected with high probability. Furthermore, epidemiological approaches in ad-hoc configurations need to address membership view issues. Probabilistic algorithms such as SCAMP [?] and LPB [?] are scalable because they require nodes to store only a small portion of global membership state. Although efficient in flat wired networks, they are not suitable for ad-hoc networks due to limited connectivity and the high cost of route resolution.

Araneola [?], a peer-to-peer system for scalable reliable multicast in highly dynamic internet environments, provided us with a framework for designing an overlay-based probabilistic message dissemination algorithm for reliable and scalable multicast in the wireless multi-hop environment. Araneola combines the best of two worlds: the resource utilization efficiency of deterministic overlay-based routing and the failure resilience and adaptability of epidemiological protocols.

Araneola’s distributed overlay creation tasks exchange messages for the purpose of cooperatively constructing a random $k/k + 1$ -regular overlay. In this overlay, each node’s degree is either k or $k + 1$ and roughly 90% of the nodes have degree k . Random k -regular graphs are expander graphs and exhibit properties that are in accordance with network protocol design goals such as k -connectivity, diameter that grows logarithmically with the number of nodes, and high connectivity after random removal of linear-sized subsets of nodes or edges [?], [?]. The first induces improved reliability through path redundancy, the second lower end-to-end delay along with efficient bandwidth utilization and the third failure resilience.

Araneola’s gossip task deterministically communicates meta-data messages over the probabilistically created overlay and actual data are then pulled from the advertising nodes. The system handles joins, leaves and failures of members with a low constant cost. Work in [?] shows that comparing to gossip protocols, deterministic flooding over a k -connected, link-minimal Harary graph induces lower message overhead (lower fanout), and higher reliability through increased connectivity. However, both [?] and [?] assume complete communication graphs and do not study the effects of hierarchical or multi-hop topologies. Our study evaluates Araneola’s deterministic flooding mechanism in ad-hoc multi-hop topologies. Our approach is to leverage the random operation of the protocol by providing each node with awareness of a portion of the network which is not confined to its physical neighborhood. To this end we use the abstraction of a unicast routing protocol.

In [?], the authors demonstrate the gains of coupling the tasks of membership view tracking and route acquisition, by introducing the *Route Driven Gossip* (RDG) protocol. RDG relies on a modified on-demand routing protocol for route and member discovery. An RDG network converges to a state in which views and route information are uniformly distributed over the nodes. We integrate RDG’s view acquisition algorithm into Araneola. This facilitates the protocol described in [?]. The altered membership protocol provides both topology and membership awareness to Araneola. Hence, more efficient network resource utilization is achieved. In addition, we perform several modifications on Araneola to render it mobility-adaptive, topology aware, and less vulnerable to the loss of connectivity.

An additional gain from unicast tunneling, is that the overlay structure does not need to change as often as the underlying topology. The change of routes among overlay neighbors is transparent to the protocol, therefore less overlay maintenance signaling is required. The unicast function however, requires additional route resolution signaling.

In this work, we proceed with a bottom-up approach, initially determining a suitable unicast routing mechanism on top of which we built a route-driven variation of Araneola. A good choice of reactive unicast wire is the Dynamic Source Routing protocol (DSR); given that it is enhanced with optimizations that aim to prevent stale routes, and reduce memory footprint and control overhead.

Although RDG relies on a reactive routing service we also evaluate a link-state protocol, in order to study the performance of proactive protocols in the investigated mode of communication. To this end, a partial-view-driven variation of Araneola is built over the Fisheye State Routing protocol (FSR) [?].

Our protocols’ implementations and simulation experiments aim at unveiling the subtleties and the tradeoffs involved. Our main contribution is evaluation of the suitability of deterministic flooding over

an expander overlay in the mobile ad-hoc environment. For this purpose, we perform numerous modifications on the base Araneola protocol to render it aware of the underlying connectivity. We compare the modified protocol to route-driven gossip and to link-level flooding.

The experimental results enable us to determine the range of ad-hoc network configurations in which deterministic overlay flooding exhibits better reliability, end-to-end delay and scalability than application layer pure-gossip schemes. In addition they provide insights on the applicability and suitability of epidemic ALM in ad-hoc settings. More specifically, our study shows that *Route Driven Araneola* (RDA) and Araneola over a link-state protocol (Araneola-LS) perform more reliably than Route Driven Gossip under low mobility, whereas in moderate to high mobility RDG outperforms Araneola-based protocols. High mobility inhibits the creation of a stable overlay geometry, both due to high loss rate of control messages and to topology changes that invalidate the established geometry.

In multicast-only networks and with all participants being receivers, flooding variations impose less signaling and yield higher reliability than the presented epidemic overlay-based protocols. Hence, our study shows that the deployment of epidemic ALM protocols for multicast-only communication is untenable. Thus, we also consider scenarios where both the unicast and multicast mode of communication are used. Under these scenarios, we show that the imposed unicast traffic amortizes the cost of application layer multicast. We also show that unicast traffic's performance degrades more under flooding than under the proposed application layer multicast function's traffic.

Organization The rest of the paper is structured as follows. Section 2 summarizes related work. Section 3 describes the base Araneola protocol. Section 4 discusses the employed unicast routing mechanisms. Section 5 describes RDG and the proposed protocols. Section 6 presents the experimentation methodology and simulation results. Sections 8 and 9 conclude and discuss future work, respectively.

2 Related Work

Below we summarize previous work related to epidemiological message dissemination and distributed construction of expander graphs.

2.1 Gossip in Wired Networks

LPBcast [?] assumes a complete (fully connected) underlying communication graph. It gossips uniformly about data messages, digests and membership state, providing reliability without imposing a complete membership view on the multicast group members. Its partial view acquisition mechanism has influenced the design of Route Driven Gossip.

Directional Gossip (DG) [?] takes into account the wide area network topology, in order to achieve efficiency gains. In short, a weight is computed for each neighbor node, representing the connectivity of the given node. The larger the weight of a node, the higher the probability for it to receive a given packet

for other nodes. When gossiping, nodes with higher weights are hence chosen with a smaller probability, reducing redundant sends. In particular, each LAN has a gossip server that is associated to internetwork routers. Two gossip servers are denoted neighbors if they are associated with the same router. Gossip servers employ bimodal gossip. They receive and inject a message from and into their local area network using a LAN gossip protocol that implies high probability of intra-LAN broadcasting. On the other hand, neighboring gossip servers interchange messages employing a distinct gossip protocol, which aims at minimizing inter-LAN traffic.

Kermarrec et al [?] present a theoretical analysis of gossip-based protocols, that relates reliability to key system parameters (system size, failure rates and fanout). They show how probabilistic gossip-based algorithms can be modeled using random graphs. Their results provide guidelines for the design of practical protocols. They demonstrate that enhancing scalability utilizing partial random views does not reduce reliability guarantees. They also show that by imposing a hierarchical structure among members, according to proximity information, less stress is imposed on the network links.

2.2 Ad-hoc Gossip

Anonymous Gossip [?] uses gossiping to improve the reliability of MAODV but assumes existing routes to gossip to any node after the delivery of a message has failed to reach all recipients. It also relies on MAODV for membership management.

Route Driven Gossip (RDG) [?] is a gossip-based multicast protocol that can be built over any on demand unicast routing service, such as DSR. RDG is described thoroughly in Section 5.1.

Gossip Based Ad-hoc Routing [?] employs per-hop gossip to reduce the flooding phase overhead of reactive ad-hoc routing protocols. It suggests a scheme where nodes relay received packets to nodes within their transmission range, with a given gossip probability. Similar to *Directional Gossip*, it proposes a two threshold scheme that retransmits with higher probability if a neighbor's degree is below a threshold. Its primary usage is the support of unicast routing protocols that utilize flooding for route discovery, such as AODV. In the context of this protocol, the bimodal behavior of gossip settings is investigated using percolation analysis and simulations. They strive to determine a value of the relay probability so that the number of executions that the message reaches almost all nodes is relatively high, while the message overhead is kept low. To this end, they provide heuristics that save up to 35% message overhead compared to flooding.

2.3 Distributed Construction Of Expander Graphs

Araneola, falls under the category of algorithms for distributed construction of expander graphs. We discuss Araneola in detail in Section 3.

Law et. al. provide an alternative solution [?]. They introduce a randomized distributed algorithm for constructing a $2d$ -regular multi-graph overlay network that consists of d Hamilton cycles. The algorithm

is completely decentralized. The constructed network is, with high probability, an expander overlay with $O(\log_d n)$ diameter. It exhibits good scalability properties because both processing and space requirements grow logarithmically. A node can join the network in $O(\log_d n)$ time with $O(\log_d n)$ messages. A node can leave in $O(1)$ time with $O(d)$ messages. In comparison, Araneola with target degree k handles joins, leaves and failures with a low constant cost of $3k$ messages. In addition, they discuss a construction of the expander network where every node can be discovered in $O(\log n)$ time. This work assumes an underlying network that provides connectivity between any pair of nodes, thereby it does not consider volatile multi-hop topologies.

GoCast [?] uses a mechanism similar to Araneola to construct a balanced node degree overlay over a wired network. Unlike Araneola, only one neighbor node is selected randomly, whereas the rest are selected strictly based on network proximity criteria. As in Araneola, the *GoCast* overlay is used for dissemination of multicast messages, however, it is complimentary to multicast over an efficient tree overlay embedded in it. The multicast messages propagate with low delay through the tree multicast tree, while overlay neighbors exchange message digests to detect and recover missing message due to failures of the tree structure. The authors claim that their approach greatly reduces the delivery delay comparing to standard gossip-based multicast systems.

3 Araneola

This section describes the Araneola system for multicast communication in a wired network [?] .

3.1 Araneola Protocol Description

Araneola constructs and maintains an approximate regular overlay in which each node's degree is k or $k + 1$, and approximately 90% of the nodes have degree k . It has been shown in [?] that a random k -regular graph $G(E, V)$, which is almost always a good expander, is a.a.s. k -connected if k satisfies

$$3 < k < |V|^{0.02} \tag{1}$$

It has also been shown [?] that $G(E, V)$'s diameter $d(G)$ a.a.s. satisfies

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

Thus, according to Equation 3.1 the system is able to achieve high reliability via path redundancy. Also according to Equation 3.1 the regular overlay provides a theoretical upper bound for end-to-end delay that grows gracefully (logarithmically) with network size.

In [?], the authors show that the protocol’s overlay structure achieves three important properties of k -regular random graphs of node size N : (a) It has $O(\log N)$ to $O(\log(N \log N))$ diameter; (b) it is generally k -connected and (c) it remains highly connected following random removal of linear-sized subset of nodes or edges. Furthermore the overlay is constructed at a low cost with each join, leave or failure to entail the transmission of only $3k$ messages in total. Emulation-based evaluation of Araneola showed that it achieves higher reliability and end-to-end delay than pure gossip protocols of up to 3 times its fanout.

The protocol described in [?] assumes a fully connected graph where each node can communicate with every other node in the Internet. However, not every node is aware of the existence of every other group member. This problem is tackled with the probabilistic membership view protocol described in [?], which provides a uniformly distributed node membership knowledge. The membership track protocol of Araneola piggybacks membership information in each gossip message in a manner similar to LPB. Each gossip message carries subscription and unsubscription information about maximum m nodes. Gossip proceeds in periodic rounds. When subscription/unsubscription entries exceed the maximum allowed numbers, the lists are randomly truncated. Views are augmented with new subscriptions and members are removed according to unsubscriptions. If the view exceeds its maximum size, then it too is randomly truncated. The removed view entries are then added in the subscription list.

Below we summarize the protocol. The interested reader can find detailed description of the protocol in [?]. The protocol maintains the following basic data structures:

- **Message Queue:** It stores the recently received messages, and the messages that are locally generated.
- **Recent IDs:** It stores the ids of the messages that were received during the previous gossip round.
- **Membership View:** It stores information about all known nodes in the overlay.
- **Neighbor Set:** It stores information about the *overlay neighbors*. Neighbor entries contain fields for the neighbor’s address, its overlay degree, and the number of gossip messages received from it during the last gossip round.
- **Missing Messages:** It stores the IDs of the messages that have been advertised but not yet received. Each *Missing Message* entry has a *Heard From* list that maintains addresses of the nodes that advertised the message.

Araneola is decomposed in three primary and two secondary tasks. The main tasks are:

Connect: This task attempts to establish neighbor relationships with randomly selected nodes in the view so that the number of overlay neighbors exceeds a low threshold *low* and does not exceed a high threshold *high*. *Low* corresponds to the target degree k . Every *connect_timeout* intervals, it sends to randomly selected nodes in the view as many *connect(degree)* messages as is the difference between the current degree and *low*. Receivers of *connect(degree)* messages respond with *connect_ok* messages only if their degree is below *high*. In this case they accept connection and update the *Neighbor Set*. If their degree is not below *high*, they redirect connection to their lowest degree neighbor.

Disconnect: This task initiates disconnections and connections among neighbors so that the degree at all nodes does not exceed *low* or *low+1* and does not drop below *low*. If a node’s n degree exceeds *low*

by a , then n selects a candidates from n 's *Neighbor Set*. If the candidate's degree is higher than low and the node has higher ID than the candidate's, then a *disconnect* message is sent. If the candidate still has a degree higher than low then it sends a *disconnect_ok* message to n and removes connection with n . Similarly, upon the reception of *disconnect_ok* message, n removes connection with the candidate.

If no candidate has greater degree than low , then the highest degree neighbor h and the lowest degree neighbor l are selected to connect to each other and to have h remove its connection with n . If $n.degree > l.degree + 2$ (to avoid exchanging messages without actually reducing the maximum degree among n, h, l) then a *connect_to(h)* message is sent to the lowest degree neighbor l . In response to this message, if $l.degree \leq low$ and if l has not handled another *connect_to* message for a duration of time equal to *connect_to_timeout*, l sends a *change_connection(n)* message to the highest degree neighbor h . If $h.degree < high$, then h initiates disconnection with n and accepts connection to l by sending a *connect_ok* message to l . When a node removes connection from an overlay neighbor it also removes all the entries in the *Heard From* lists that correspond to the disconnected neighbor's address.

Gossip: This task sends *gossip* messages every *gossip_timeout* to the overlay neighbors. The *gossips* contain last round's *Recent IDs* (digests). The neighbors that receive the message digests place the address of the advertisers in the end of the corresponding *Missing Message's Heard From* list. The *gossip* messages also carry requests for any possible missing messages for which the first entry in the *Heard From* list is the intended recipient of the *gossip* message.

When a node receives a request it explicitly replies with the associated *data* message. Upon requesting a missing message, the task moves the corresponding first entry of the missing message's *Heard From* list to the end of the list. In this way, if the missing message is not delivered by the node it was last requested, it will be requested by a node that has more recently gossiped its existence. This considers the fact that the node that failed to deliver the *data* message at the first place, is likely to fail delivering it again, since the reasons of the failure may still be present (node failure, network partition). Upon reception of a missing *data* message, a node removes the associated *Missing Message* entry.

The secondary tasks are *Failure Detection* and *Garbage Collection*. Failure detection periodically checks the *heartbeat* field of the neighbor entries and if it is below a minimum threshold, then connection to this neighbor is terminated. Garbage collection simply cleans *Message Queue* and *Missing Messages* from old entries.

4 Ad-hoc Routing Protocols

In this section we present an overview of the ad-hoc routing techniques employed by our protocols.

4.1 Dynamic Source Routing

4.1.1 DSR Overview

DSR is a reactive (or on-demand) ad-hoc routing protocol that employs source routing and aggressive route caching. Routes are resolved by flooding requests and source routed replies. The route discovery phase yields redundant routes to a destination because route destinations reply to all received requests. In the process, intermediate nodes in the reply path also resolve routes to this destination. If backward learning is enabled, assuming symmetric links, reversed routes are resolved upon reception of a request.

Source routing enables DSR to detect loops and to acquire topological information by promiscuously listening to next-hop nodes transmissions. DSR assumes link layer failure feedback from the MAC layer. It uses this feedback, to propagate route failure notifications to nodes in the upstream. The on-demand nature of the protocol eliminates the need for periodic updates and neighbor discovery beacons. The DSR version we use includes several optimizations proposed in [?].¹

In [?] the authors observe a serious design flaw: there is no effective mechanism for expiring stale routes or selecting among the fresher routes. This results in DSR incurring routing failures in the presence of frequent topological changes. To address this issue, we augment DSR route cache entries with timestamps, which are needed to implement a lifetime-based cache eviction policy. In [?] we present an analysis for optimizing the lifetime of route cache entries with respect to routing delay. The optimal lifetime is computed with respect to the average node speed, the transmission range, and the hop count of the route.

4.2 Fisheye State Routing

Fisheye State Routing (FSR) is a proactive protocol. These protocols are also known as state-based/table-driven protocols. Protocols that fall in this category perform periodic route table exchanges and continuously attempt to maintain a complete topological view of the network at each node. Hence, routes are readily available when needed.

FSR is a link-state protocol. To reduce the overhead of periodic link state packets, FSR modifies the link-state algorithm in the following three ways: (a) link-state packets are no longer flooded. Instead, only neighboring nodes exchange the topology table information; (b) the link-state exchange is solely time triggered and not event triggered; (c) instead of periodically transmitting the entire link-state information, FSR uses different exchange intervals for different types of entries in the topology table. Link-state entries corresponding to nodes that are within a predefined distance (scope) are propagated to neighbors more frequently (intra updates) than entries of nodes that lie outside the scope (inter updates).

With scoped flooding nodes maintain less accurate topology information on distant network regions.

¹ We list these optimizations in [?].

However, as a packet is forwarded toward a destination, nodes have increasingly accurate routing information, resulting to correct forwarding. Owing to these modifications, FSR scales well.

There are four configuration parameters for FSR, the value of which depends on factors such as mobility, node density and transmission range. We list them below.

- *Size of the scope*: This parameter specifies the scope radius of a node in number of hops.
- *Time out for the neighboring nodes*: If a node does not hear from a neighbor specified by this value, the neighbor node will be deleted from the neighbor list.
- *Intra scope update interval*: The update interval of sending the updates of the nodes within the scope radius.
- *Inter scope update interval*: The update interval of sending the updates of the nodes outside the scope radius.

5 Araneola-based Ad-hoc Deterministic Flooding

In this section we explore the idea of building an Araneola-based application layer deterministic flooding scheme. We use a unicast protocol to provide abstraction of higher than link-level connectivity. In doing this we increase the size of a membership view, leveraging the randomness of the regular overlay geometry, and thereby its “good” properties.

In ad-hoc networks, route information that is obtained reactively is valuable because of the cost of flood-based route discovery. There are no connectivity guarantees for every pair of nodes in the network. Therefore, a random membership tracking mechanism, which is aware of the route state is more appropriate than LPB’s view-driven approach. This mechanism does not trigger route requests for every destination that is selected to be an overlay neighbor. On the contrary, it is the same fact that the route to the destination is already available that causes the destination to be included in the view. The membership protocol of [?] reduces the route discovery overhead of reactive unicast routing, while it provides a random partial view that facilitates the process of creating the random regular overlay over the ad-hoc network. Furthermore, its flood-based group-request phase constitutes an effective way for acquiring membership information, in the absence of initial partial view knowledge.

In addition, the proposed overlay deterministic flooding scheme needs to adapt to the volatile multi-hop topology. In Section 5.2 we present modifications of the base protocol that render it more aware of topology, group member reachability and of the fact that mobility introduces temporary link failures.

5.1 Route Driven Gossip

This section describes Route Driven Gossip (RDG) [?] and its route-driven membership management mechanism. RDG is a gossip-based multicast protocol that can be built over any on-demand unicast routing service. It departs from the view-driven model, in which the nodes select the gossip partners from their membership view. In ad-hoc environments there is no guarantee that a node can reach a host

in its view. Route driven means that the protocol relies on views that are randomly acquired via the route discovery process. A node gossips only to the nodes for which the routes are known.

RDG maintains the following basic data structures:

- **Active View** contains the IDs of known members, to which at least some route is resolved
- **Passive View** contains the IDs of known members to which no route is available.
- **Remove View** contains the IDs of known members that have inserted a leave request.
- **Data Buffer** stores data messages received. It is divided in *Buffer.new* which contains the messages to be gossiped in the future and *Buffer.old* which contains old messages for responding to gossip pulls(see below).

The sets of active view, passive view and remove view members are denoted AV , PV and RV respectively. The gossip fanout is denoted as F . All data structures have a maximum size. When views reach their maximum size they are randomly truncated.

RDG extends the message set of DSR with the *GroupRequest* and *GroupReply* primitives. These primitives are used to establish routes to group members with a single request flooding.

RDG has seven protocol operations that are classified into three sessions. The *join* session involves the behavior of a joining node and of the nodes that react to its join request. The *gossip* session involves the periodical dissemination of messages by nodes and the reaction to these messages by the receivers. The *leave* session involves the interactions that take place when a node wishes to leave the network. Since the *gossip* and *leave* session are tightly coupled (the leave request is carried in gossip messages), we describe them together.

The **join** session has as follows:

- (a) Upon join, a node floods a *GroupRequest* message to initiate route discovery and announce its membership.
- (b) Upon receiving a *GroupRequest*, all nodes augment their AV with the initiator's ID. They reply to the requester with probability p .
- (c) The initiator augments its AV with the replier's ID.

The **gossip/leave** session has as follows:

- (a) With a period T_g the gossip task initiates gossip packet transmission to F randomly selected active view members. Each *Gossip* packet carries at most *data_messages_per_packet* message digests (IDs) of data messages in *Buffer.new*, and the ID of the most recent missing data message. A missing message is detected with out-of-order receptions (data messages are assigned a sequence number) and is requested if it remains missing for m gossip rounds. The *Gossip* packet also carries view information. V randomly selected members from $AV \cup PV$ are added to the view field of the message. Also, r random members

of RV are added to the gossip's remove view fields. If the node intends to leave the network, it only includes its ID in the remove view field. A message in $Buffer.new$ is placed in $Buffer.old$ after it has been gossiped t_g times.

(b) Upon reception of a *Gossip* packet the new view members are added to AV or PV according to the route state. E.g if a route to the new view member exists, it is added in AV . The ID in the gossip remove view field is removed from the AV or PV view and is added in RV .

(c) When a node receives a *Gossip* packet it scans the message digests for missing messages. Then, it immediately sends a *DataRequest* packet that contains the list of the missing messages to the sender of the *Gossip* packet. Finally, the node responds to the requested missing data message with a *GossipResponse* packet, only if this message is in $Buffer.old$.

(d) Upon reception of a *DataRequest* packet a node immediately replies to the sender with a *Data* packet. The *Data* packet contains the data messages requested.

(e) Data messages that are received in *Data* and *GossipResponse* messages are delivered to the upper layer only if the message is still missing.

(f) If $|AV|$, $|BV|$ or $|RV|$ exceed a predefined threshold, a random member is removed. AV and BV are periodically checked for the consistency of membership with members' route state and are adjusted accordingly.

The above view gossip and join mechanisms induce uniform distribution of views and routes in the network.

RDG is rendered topology aware by assigning weights to the active view routes. The weights are inversely proportional to the route's hop count. Active view members with high weights are selected with higher probability to become gossip destinations. In this way, nodes gossip to members in their proximity, reducing the number of hops that gossip messages traverse. The weighted view is thoroughly described in Section 5.6. It is shown in [?] that topology awareness yields improved reliability and end-to-end delay.

Intermediate nodes in the routing path to gossip destinations are promiscuous receivers of messages that are being forwarded. The mechanism is similar to the one described in detail in Section 5.3.

As proposed in [?], the binary exponential backoff algorithm is employed for adjusting the gossip period when there is no message to be sent or to be requested.

5.2 Route Driven Araneola

By combining the membership view management of RDG with the random regular overlay creation and message dissemination mechanism of Araneola, we build a route-driven variation of deterministic

overlay flooding, namely *Route Driven Araneola (RDA)*.

In designing RDA, we first address the issue of selecting the most appropriate reactive unicast abstraction. In [?], we performed evaluations of unicast protocols to determine the most suitable for the mode of communication we are interested in: point to multipoint communication, low to moderate mobility and moderate traffic load. The evaluation compared optimized AODV with a modified version of DSR with lifetime regulated route cache entries and some of the optimizations proposed in [?]. Our optimized DSR version demonstrated reliability and control overhead marginally superior to AODV, for low mobility and moderate packet injection rate [?]. In these configurations, lifetime-regulated redundant route entries provide additional valid routes reducing route discovery overhead.

Fig. 5.2 gives a representative architectural description of ad-hoc multicast using Araneola. Fig. 5.3(a) lists the data structures and variables used in the pseudocode descriptions of RDA. Fig. 5.3(b) describes the Araneola *connect* task, including the modifications presented in the rest of this section.

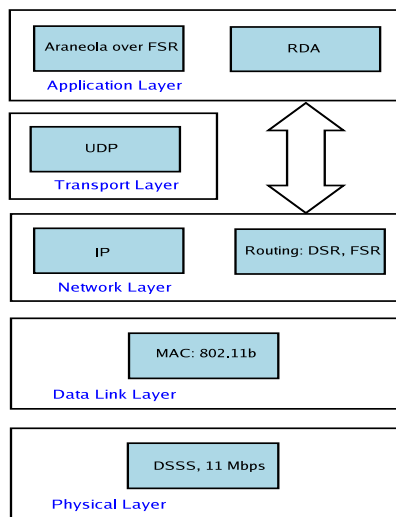


Fig. 1. Layered architecture of Araneola multicast in the ad-hoc environment.

5.3 Membership Management and Interaction with DSR

In order for DSR to provide RDA with view acquisition and route awareness, we augment DSR in a manner similar to *Route Driven Gossip*:

(a) We add the *InitiateGroupRequest()* interface, which DSR exports to Araneola. Araneola invokes it when the node joins the network or when the size of the *active view* drops below a specified threshold.

(b) A *GroupRequest(groupId, sourceAddress)* message is flooded upon invocation of *InitiateGroupRequest()*. Optionally, this request can have small TTL to restrict the members of the *active view* only to nodes in the proximity. Hence, small TTL renders the protocol topology aware by reducing the hop count toward unicast destinations. It also prevents group request broadcast storms. The downside is its impact on the randomness of the k-regular overlay creation.

(c) We define the *RegisterRouteUpdateNotifyCallBack()* interface. DSR exports it to Araneola for callback registration. This callback is called each time a route to a new destination is acquired via a *Group Request* message handled by the *ReceiveGroupRequest* function (see below). It is also called when routes to a destination cease to exist. The latter occurs when the only route entry for a destination is expired or one of its link is reported broken by a DSR *Route Error* message.

(d) We define DSR's *ReceiveGroupRequest()* function, which initiates, with a specified probability, a *Route Reply* in response to *Group Request* messages. To avoid broadcast storms and to enforce higher degree of DSR route path disjointness, duplicate received *Group Requests* are not further relayed. *Group Reply* messages are sent only when the group ID of the *Group Request* matches the node's group ID. Reception of a new request results in acquisition of a new route.

(e) DSR exports the *CheckRouteExist()* and *GetHopCount()* functions to RDA. The former is used for determining whether a route to a specific destination is available. The latter is used to obtain the hop count to it, thus enabling topology awareness.

We provide pseudocode description for the above in Fig. 5.3(a).

In developing Route Driven Araneola (RDA), we augment the base Araneola protocol to support partial view acquisition and interaction with DSR. Fig. 5.3(b) provides pseudocode description of the modifications: :

(a) We partition a view into *Active*, *Passive* and *Remove View*.

(b) At startup, RDA registers a *RouteUpdateNotify()* callback with DSR. This callback notifies RDA every time a destination becomes reachable or unreachable via the DSR routing function. RDA updates the view information of the destination accordingly. In case a destination is reported unreachable, while it is an overlay neighbor, it is not removed from the active view. Since the destination remains connected, it is likely that in the next gossip round a new route will be resolved. We expand on this issue later in the Section 5.4. Furthermore, no action is taken when the path to a non-view-member node ceases to exist.

(c) Upon startup, RDA invokes DSR's *InitiateRouteRequest()*. Receivers of *GroupRequest* and *GroupReply* messages update their active views by calling the *RouteUpdateNotify()* callback upon route establishment. *InitiateGroupRequest()* is also called when the *active view* size drops below a specified threshold.

(d) A periodic task (*CheckViewAndRouteState*) checks the route state of nodes in the view and adjusts their view according to route availability. If the node in the view is an overlay neighbor and it becomes unreachable, the task does not change its view state.

(e) We augment the *gossip* message with random view information similar to the RDG algorithm. V randomly selected members from $AV \cup PV$ are added to the view field of the message. Also, r random members of RV are added to the gossip's remove view fields. If the node intends to leave the network, it

Data structures:

id - a node's identifier.
nid - this node's identifier.
gid - the identifier of the group.
clock - the current time.
degree - this node's out-degree.
neighbors - set of pairs (*id*, *degree*), initially \emptyset .
mid - a message's identifier, pair of (sequence number, source *id*).
message - pair of (*mid*, data).
messages - queue of *message*'s, initially \emptyset .
heard_from - queue of nodes.
missing_messages - set of pairs (*mid*, *heard_from*) initially \emptyset .
recent_mids - set of *mids*'s, initially \emptyset .
AView - active view, set of *id*'s.
PView - passive view, set of *id*'s.
RView - remove view, set of *id*'s.

Parameters:

low - target number of overlay neighbors (degree).
high - upper bound on the number of neighbors.
connect_timeout - a time value.
maxAV - maximum entries in *AView*.
maxPV - maximum entries in *PView*.
maxRV - maximum entries in *RView*.

```

01. RDA_Connect()
02.  diff = low - |neighbors|
03.  while (diff > 0)
04.    candidate = random node from AView
05.    RDA_Send(CONNECT, neighbors) to candidate
06.    schedule RDA_Connect() after connect_timeout

08. RDA_ReceiveConnect(node_id source, int degree)
09.  if (|neighbors| < high)
10.    RDA_AddConnection(source, degree, true)
11.  else
12.    RDA_Send(REDIRECT) to lowest degree neighbor

13. RDA_AddConnection(node_id source, int degree, boolean ack)
14.  AView = AView  $\cup$  source
15.  if (|AView| > maxAV)
16.    remove random member from AView
17.  neighbors = neighbors  $\cup$  {source, degree,
18.    min_heartbeat}
19.  if (ack == true)
20.    RDA_Send(CONNECT_OK, |neighbors|) to source

20. RDA_ReceiveRedirect(node_id source, node_id redirect_to)
21.  if (DSR_RouteExists(redirect_to))
22.    AView = AView  $\cup$  redirect_to
23.    if (|AView| > maxAV)
24.      remove random member from AView
25.      RDA_Send(CONNECT, |neighbors|) to redirect_to
26.    else
27.      PView = PView  $\cup$  redirect_to
28.      if (|PView| > maxPV)
29.        remove random member from PView
30.      candidate = random node from AView
31.      RDA_Send(CONNECT, neighbors) to candidate

32. RDA_ReceiveConnectOk(node_id source, int degree)
33.  if (|neighbors| < high)
34.    RDA_AddConnection(source, degree, false)
35.  else
36.    RDA_Send(LEAVE) to source

37. RDA_ReceiveLeave(node_id source)
38.  RDA_RemoveConnection(source)

39. RDA_RemoveConnection(node_id n)
40.  remove n from neighbors
41.  if (|neighbors| < low)
42.    RDA_Connect()

```

(a)

(b)

Fig. 2. (a) Data structures and parameters of RDA. (b) RDA Connect Task. The modified Araneola overlay construction.

only includes its ID in the remove view field

(f) RDA registers a promiscuous listening function with DSR. This callback is used by DSR's *router* function. It serves the purpose of acquiring knowledge of missing message ID's and receiving missing messages, regardless of whether the message is destined for or forwarded by the receiving node. Promiscuously received missing messages and ID's are placed in the respective data structures as if they were regularly acquired. For example, a node can learn for a missing message by eavesdropping a *gossip* message and later receive it by eavesdropping the subsequent data message. In case the subsequent message is not received, the node can request it from the sender as soon the sender becomes its neighbor or by performing a repair pull (see below). Nodes still gossip about messages that were received through this mechanism.

(g) Each time a new member is acquired by the gossip task, it calls DSR's *CheckRouteExist()* to determine whether the new member should be placed in the *Active* or *Passive* view.

(h) As mentioned in Section 5.1, when the size of a view exceeds a predefined threshold, the view

| | |
|---|--|
| <pre> 01. DSR_InitiateGroupRequest(node_id gid) 02. fbod (GROUP_REQUEST, gid, nid) 03. DSR_ReceiveGroupRequest(node_id source, node_id gid) 04. if (source belongs to group with identifier gid) 05. RDA_RouteUpdateNotify(source, true) 06. with a certain probability send(GROUP_REPLY) to source) 07. DSR_ReceiveGroupReply(node_id source) 08. RDA_RouteUpdateNotify(source, true) 09. DSR_ReceiveRouteError(node_id unreachable_node) 10. RDA_RouteUpdateNotify(unreachable_node, false) 11. DSR_RouteExists(id) 12. if route to id exists) 13. return true 14. else 15. return false </pre> | <pre> 01. RDA_Join(node_id gid) 02. DSR_InitiateGroupRequest(gid) 03. RDA_RouteUpdateNotify(node_id id, isRouteAcquired) 04. if (isRouteAcquired) 05. AView = AView ∪ id 06. if (AView > maxAV) 07. remove random member from AView 08. else 09. if (id ∉ neighbors) 10. PView = PView ∪ id 11. if (PView > maxPV) 12. remove random member from PView 13. RDA_CheckViewAndRouteState() 14. for all members i in AView 15. if (DSR_CheckRouteExists(i) == false) 16. AView = AView \ id 17. if (PView > maxPV) 18. remove random member from PView 19. PView = PView ∪ id 20. else 21. PView = PView \ id 22. if (AView > maxAV) 23. remove random member from AView 24. AView = AView ∪ id </pre> |
| (a) | (b) |

Fig. 3. (a) DSR interface for supporting group join and route state. (b) RDA join initiation, route state update notification and view management.

is randomly truncated. Only members of the active view, which are not overlay neighbors, can be removed. The removed member is added to the view of the next gossip message.

(i) *Remove View* entries are expired after a specified interval to prevent them from remaining in the system indefinitely [?]. For this purpose *Remove View* entries are associated with timestamps. *Active* and *Passive View* members are not expired in order to ensure uniformly distributed views.

5.4 Route Update Notification Policy

We considered three route update notification policies:

(a) DSR notifies Araneola upon every acquisition of a new route, regardless the way it was resolved (reply, group reply, gratuitous reply, routes to intermediate nodes, backward learning). This induces locality of view information; nodes in the proximity are included in the view with higher probability. However, it significantly increases the size of the view, leveraging randomness in the overlay construction. In addition, gossiping to nodes in the proximity reduces latency and increases reliability.

(b) DSR notifies Araneola only upon the reception of replies to group requests and reception of group requests (lines 3-15 in Fig. 5.3(a). View acquisition is mainly driven by the join process and the gossip task. Routes are still acquired with the above mentioned mechanisms. Through the gossip task and a periodic route state check task, active view can be populated with additional nodes to which routes are already resolved. There is locality of membership information for the reasons mentioned in (a). The increased view size and the degree of randomness introduced by the group request and gossip phase are in accordance with the random overlay goal.

(c) DSR notifies Araneola only upon the reception of replies to group requests and the reception of a group request. However, routes are acquired solely through the group request/reply process and no aggressive route acquisition policy is applied. Since all route acquisitions are reported to Araneola, active view is not modified through the gossip task, therefore view dissemination in gossip messages is disabled. View distribution is independent and identical, although its size is reduced and is not updated regularly.

In all alternatives, DSR always notifies Araneola when a destination in the route cache ceases to be reachable. Recall that if the destination is an overlay neighbor then it is not removed from the active view.

We choose to implement choice (b), which represents a compromise between approaches (a) and (c). It maintains a degree of view uniformity, while it populates the view at a faster rate. Nodes are added in the *Active* view, when connection requests are received from them. DSR backward learning ensures that the reverse route to the node that initiated connection, is already resolved. Even if there is no route to the node that initiated connection, it will eventually be resolved during the subsequent gossip round.

DSR perceives all packet drops as being caused by route failures. Consequently, drops due to collisions result in a cached route to be deleted and the destination member to be incorrectly placed in the passive view. In the absence of a way to distinguish packet collisions from link failures in the MAC or network/DSR layer, we consider this inaccuracy unavoidable.

5.5 Membership and Neighbor Tracking Issues

Optionally, we can enhance view uniformity to ensure that all members are equally known. This is achieved by assigning weights to view members [?]. Each time an existing member is gossiped, the weight of that entry is increased. During the gossip phase members with lower weights have higher probability to be gossiped. When views reach maximum size, they are truncated removing members of higher weights. In fact, we have not implemented this feature because in the small scale networks that we are experimenting with, in order to ensure high randomness we had to maintain views almost a third of the size of the network. Therefore each node is with high probability "well known" in the network.

When a node receives a connection initiation message by a non-member, the sender is automatically added in the active view, since it is likely that the reversed path has been resolved too. Furthermore, we consider two *membership* and *overlay neighborhood tracking policies*. Due to these policies and the fact that overlay neighbors are not put in passive view upon becoming unreachable, all overlay neighbors are always active view members. Both policies imply modifications to the base Araneola protocol . The two policies are:

(a) Add Gossip Sender in Active View: As noted above, when a route to an overlay neighbor ceases to exist, this node remains in the active view. Consequently, if in the next gossip round, a route has not already been resolved, a flood-based route-request will be generated. However no overlay maintenance overhead is induced.

Due to the frequent loss of connectivity, inconsistencies in overlay state among neighbors are common. We have determined that these inconsistencies have a degrading effect on the overlay regularity and result in packet losses and high end-to-end delays. To tackle this problem, we modify the algorithm that is described in Section 3 as follows.

The receiver, *dest*, of a *gossip* message, *gm*, from a non-overlay neighbor, *src*, instead of ignoring *gm*, it acts as follows. If the degree of *dest* does not exceed the *high* threshold, then *dest* adds *src* to its neighbor set and adds *src* to the active view, regardless of view state. Otherwise, *dest* sends a *leave* to the *src*.

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

(b) Connect to reachable only: The base algorithm changes so that when a gossip message from a non-neighbor is received, the node adds the sender in its *active* or *passive view* according to reachability. If the sender is reachable, and the degree is below *high* threshold, a connection is added, otherwise a leave message is sent to the sender. If the sender is not reachable, then the message is ignored.

As an optimization that prevents route discovery flooding, 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 a randomly selected active node, and if it is already in its membership view it adds the indicated node to the passive view (lines 21-32 in Fig. 5.3(b)). In the case of *connect_to* message though (see Section 3, the receiver has to send a *change_connection* message to the designated node so that the designated node removes its connection with the sender of the *connect_to* message. The receiver of the *change_connection* message, in its turn, if its degree is below *high*, must accept the connection and reply with *connect_ok*, regardless of the route state. In Fig. 5.3(b) we describe the protocol only for policy (b).

5.6 Protocol Fine-tuning and Optimizations

Unlike Araneola, RDA does not issue a data message in distinct packets in response of a data request in *gossip* messages. Instead, in order to reduce the byte and processing overhead associated with transmitting a packet it uses a technique reminiscent of Nagle's algorithm for TCP. It waits for a specified interval to accumulate a certain number of requested messages, which it bundles in a single *data* packet. Each *data* message contains a predetermined maximum number of data messages.

The *gossip_timeout* parameter (defined in Section 3 is set constant to the number of messages per data packet (m_{pdp}) seconds. This is an adjustment for the base case of 1 msg/sec. However, if the rate is lower, data packets in response of requests will simply carry less than m_{pdp} messages. If the rate is higher, more than one data packets will be sent to a specific requester during a gossip round.

The base Araneola protocol assumes an environment where node and edge failures are likely to be persistent (e.g. a permanent process failure in a LAN). Hence, it is logical to remove any state stored

in the network regarding the failed member. However, in the ad-hoc network case, this does not hold because a node that becomes unreachable and is removed from the neighbor set can become reachable and come into the neighbor view at a later time. Thereby, we modified base Araneola so that it does not remove entries from the *heard_from* lists of missing messages when a connection is removed (line 42 in Fig. 5.3(b)). In this way the message can still be requested from its advertiser, when a new overlay connection is later established. Although this modification significantly increases maximum end-to-end delay, it improves reliability.

To provide higher reliability, we enhance ad-hoc Araneola with an *Error Repair Data-pull* phase. If a message m remains missing at node p for a specified period of time,² then p initiates a *data request* with an explicit *gossip* message to the node that corresponds to the first entry in the *heard_from* list of m . p then increases a counter. If the counter reaches a threshold, m is considered not available and the m 's entry is deleted. Note that since we keep *heard_from* entries of removed neighbors, the *data request* may be sent to a non-overlay neighbor. In this case, unlike the two policies described above, the request receiver does not attempt to establish Araneola connection with p .

The *Connect* and *Disconnect* tasks need to adapt to mobility and evolving connectivity. As changes in connectivity become more frequent, the maintenance tasks should be awakened more frequently in order to provide the RDA network with more updated overlay state information. Furthermore, mobility encumbers overlay connection establishment by invalidating routes and causing the loss of RDA control messages. RDA should cope with mobility by increasing the target degree, allowing it to approximate the desired overlay degree. Consequently, overlay maintenance and gossip timeouts, as well as the *high* and *low* thresholds are set with respect to mobility. These parameters have been empirically obtained (see Table 3). We observed that the choice of *low* and *high* threshold, has substantial impact on the performance of the protocol.

Efficiency in terms of network and power resources is critical. Since the overlay edges are not selected according to link cost and data delivery delay criteria, resources are not optimally utilized. A major challenge is optimal selection of overlay neighbors to reduce communication cost, while maintaining the random properties of the overlay. RDA can become *topology aware* using a technique similar to the weighted active view employed by RDG. To enhance view adaptation to topology we define the probability p_i that a node i at hop distance h_i is selected from the active view, to be proportional to the probability of successful transmission to this node. The probability of successful transmission to the next hop is denoted p_s .

$$p_i = \frac{p_s^{h_i}}{\sum_{i \text{ in } AV} p_s^{h_i}} \quad (3)$$

The *Connect* task selects view members to send connection requests (line 4 in Fig. 5.3(b)), according to the probabilities p_i . This approach potentially affects the randomness of the regular graph, increasing

² Normally, the missing message TTL is equal to the maximum predicted end-to-end delay which is two times the maximum overlay diameter times the *gossip_timeout*.

diameter and causing uneven degree distribution. However, the mobility model mitigates this effect by randomly distributing distances among nodes.

5.7 RDA Overlay

Fig. 4 shows a snapshot of the ad-hoc network during a simulation run that visualizes the regular overlay for a small subset of the RDA network. For this figure we used a topology aware and mobility adaptive variation. Notably, nodes select as neighbors nodes in the proximity and the node degree approximates the target five. In RDA the common case is that paths are resolved for all neighbors. We also observe that in the case of node 5, which is located in a less crowded region of the network, routes toward neighbors overlap significantly. Node 80 however, which resides in a crowded region, connects to its neighbors via routes in a star-like manner. The first case may result in saturating the common portions of the path, while the second yields less congested routes. Node 5’s active view consisted of nodes connected via a single path, while 80 has a larger selection of connection candidates. Furthermore, 5 is more likely to disconnect from the overlay network because a single link failure can interrupt communication with all its overlay neighbors.

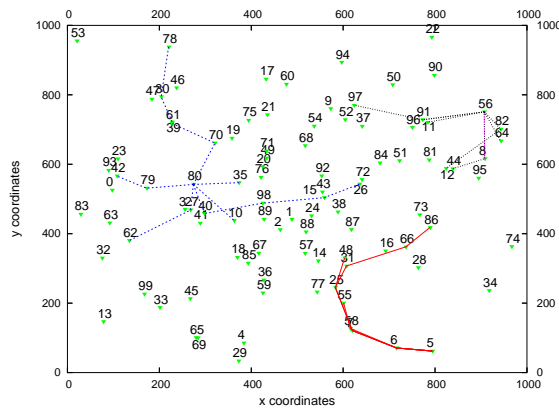


Fig. 4. Topology aware and mobility adaptive RDA network snapshot after 200 seconds of simulation. Average node speed is 1 m/s and 1 multicast sender injects 1 msg/s. The figure shows the routes toward overlay neighbors of nodes 5, 56 and 80. Routes of the same node are depicted with the same colour. Green dots represent mobile nodes. Neighbors of 5 are 6, 7, 48, 55 and 86. Neighbors of 56 are 8, 11, 12, 44, 64, 91, and 97. Neighbors of 80 are 10, 26, 35, 42, 62 and 78.

5.8 Araneola with Ad-hoc Link-state Routing

We also considered Araneola over a proactive routing protocol, namely Araneola-LS. Araneola-LS utilizes Fisheye State Routing and the LPB membership tracking algorithm. Araneola-LS makes use of the base Araneola protocol and does not determine the membership view or the overlay connectivity according to reachability. However, it incorporates all the optimizations described in Section 5.6. It is promiscuous through IP layer’s routing function and topology aware using hop metrics obtained from FSR’s topology table.

6 Experimentation Methodology

We present a simulation evaluation that considers the evolving per-hop connectivity and the effects of MAC layer interactions, such as contention-induced frame losses. Through our simulations we are able to quantify the protocols' reliability, expected end-to-end delay and control overhead. Although, in the early stages of our research we implemented and experimented with a simple model for node joins/leave [?], our study does not focus on this type of churn. We wish to focus our study in mobility-induced churn, which causes temporary link failures. Mobility by itself suffices in creating a highly dynamic network environment.

Below we describe the simulated system architecture, the experimentation methodology, and the simulation parameter setting.

6.1 Glomosim

We use the Glomosim [?] parallel discrete-event simulation library for wireless networks. *RDA*, *RDG*, and *Araneola-LS* are implemented in the application layer. The simulated application layer protocols do not rely on transport layer reliability guarantees, therefore they utilize UDP as a thin transport protocol.

The Glomosim distribution included DSR, which is implemented in the network layer as a complementary source routing function. DSR was modified as discussed in Section 4.1.1. FSR is provided in the application layer and carries out its functionality via updates of the IP forwarding table. At the MAC layer we use the 802.11 Distributed Coordination Function, which provides controlled access to the medium using virtual and physical carrier sense mechanisms for collision avoidance. All correctly received unicast transmissions are acknowledged by the 802.11 receivers.

To simulate a realistic radio environment, we employ statistical propagation models for multipath fading [?]. These models account for signal strength loss that occurs when the signal is received out of phase from multiple paths due to reflections. For the large scale fading we use the two ray propagation model and for the small scale we employ the Ricean fading model. The radio characteristics of transmitters and the nominal bandwidth value are listed in Table 1. They are configured to closely match characteristics of commercial 802.11b devices. The listed radio settings yield an average effective transmission range of 149.623 m.

6.2 Methodology

For the purpose of this study, we are evaluating the protocols with a single group consisting of all the nodes in the network. Transition to a multigroup network is trivial.

For the multicast protocols, we perform comparative evaluation with one sender and varying mobility. Also, the multicast protocols are compared with both multicast and unicast traffic (mixed traffic), generated by a single multicast sender and multiple unicast sources. The number of unicast sources is varying.

| | |
|---------------------|---------|
| Transmission power | 19 dbm |
| Antenna gain | 12 db |
| Antenna height | 0.1 m |
| Radio sensitivity | -84 dbm |
| Operation frequency | 2.4 GHz |
| Bandwidth | 11 Mbps |

Table 1

Technical characteristics of simulated transmitters.

Since there is a single multicast sender, load balancing is affected; nodes closer to the multicast source incur increased overhead. To alleviate this effect, the Araneola protocol is modified so that the source does not gossip consistently over the overlay, but to nodes randomly selected on a per-round basis [?]. Random mobility also induces load balancing.

We implement a simple CBR traffic generator for the multicast protocols. Upon simulation initialization, the selected nodes initiate the generation of messages in specified intervals. The traffic load characteristics are: a) 512 byte data message size and b) rate equal to one message per second.

The average speed ranges from 0 to 20 m/s. For no mobility, we place a hundred nodes in 100 m intervals over a $950\text{m} \times 950\text{m}$ grid. For average speed in the range from 1-20m/s, we initially place nodes randomly over a $1000 \times 1000 \text{ m}^2$ terrain. We utilize the popular random waypoint mobility model [?] with 5 seconds pause time. In order to mitigate this model's inherent problem of converging to lower average speeds [?], the difference between minimum and maximum speed is only 2 m/s.

The average number of physical neighbors is 7.94.³ Each graph point corresponds to the statistical mean obtained by 20 runs with varying random generator seeds. 95% confidence intervals are also estimated.⁴

The simulation duration is set to 600 s. To allow enough time for more messages to reach the group, member sources generate load until the 560th second of the simulation. The maximum size of the application layer PDU is 2176 bytes, therefore the RDA or RDG data packet accommodates at most four 512-byte data messages and the application layer header.

We use the following metrics to evaluate the protocols [?]:

- (1) *Reliability*. It corresponds to the data message delivery ratio. This ratio is the number of data messages delivered to destinations divided by the number of data messages sent by the source, times the number of destinations.
- (2) *Control Overhead over Delivered Data Bytes*. It is the network-wide number of control bytes transmitted over data bytes delivered to the destination application. It measures the efficiency of the protocol in expending control overhead to deliver data. Note that this does not includes only the

³ As measured with neighbor beacons transmitted every 2 s at 2 m/s average speed.

⁴ For reasons of readability, they are not depicted in all figures.

bits in the routing control packets, but also the bits in the header of the data packets, underlying protocol control packets (802.11 RTS/CTS/ACKS) and header overhead. It depends on hop count and reliability. The more efficiently the protocol utilizes bandwidth, the more nodes can be supported in the network, therefore this metric is an indicator of the protocol's scalability. We also refer to it as *delivery overhead*.

- (3) *Control Overhead over Transmitted Data Bytes*. It is the network-wide number of control bytes transmitted over data bytes transmitted. It measures the efficiency of the protocol in expending control overhead to transmit data. It is not skewed by hop count and reliability. Similar to the delivery overhead, this metric is an indicator of scalability. We also refer to it as *transmission overhead*.
- (4) *Average and maximum end-to-end delay (sec)*. They include processing and queuing delay at the intermediate nodes and are a better indicator of latency than hop count. These metrics assist us in inferring the overlay diameter.

We modified the 802.11 glomosim code to distinguish control and data bytes at the the MAC layer, by examining the 802.11, UDP and application layer headers. In addition we use the following metrics as indicators of the RDA overlay properties:

- (1) *Diameter*. It corresponds to the maximum overlay hop count. It can also be inferred by the maximum latency observed in the network.
- (2) *Average Degree*. The upper bound of a non-regular graph's connectivity is its minimum degree. k -connectivity means that the removal of any $k - 1$ sized subset of nodes or edges will not cause graph partition, but there is at least a subset of size k that disconnects the graph. We use the average over time and nodes degree metric to approximate overlay connectivity.
- (3) *Average Connection Duration (s)*. By measuring the the average over time and nodes duration of connection relations between group members, we are able to determine the stability of the random regular connection graph.

6.3 Parameters

In the presence of mobility, the protocols need to adapt to changing topology dynamics. Hence, their simulation parameters vary according to average speed of nodes and transmission range.

DSR: Initially, *Route Requests* are sent with TTL equal to 1. If a node does not receive a reply within 500 ms, it is retransmitted with TTL set to a maximum value. The request timeout is then set to 30 ms and is doubled with every repeated attempt. An entry is removed from the *Route Request Seen* table after 30 s. The lifetime of route cache entries is calculated according to equations presented in [?]

FSR: The neighbor timeout interval I is empirically determined through simulation for optimizing reliability. These values, with respect to mobility, are shown in Table6.3. Intra scope update interval equals I and inter scope update interval is set to 3 times I . The scope is 2.

| Average speed (m/s) | Neighbor timeout (sec) |
|---------------------|------------------------|
| 0-4 | 5 |
| 5-9 | 4 |
| 10-15 | 3 |
| 16-20 | 2 |

Table 2
Fisheye neighbor timeout with respect to mobility

Ad-hoc Araneola: Table 3 lists the parameters of mobility-adaptive RDA and ad-hoc Araneola over link-state routing with respect to average node speed.

| Parameters | 0 m/s | 1 m/s | 2 m/s | 3-4 m/s | 5-6 m/s | 7-10 m/s | 11-15 m/s | 16-20 m/s |
|--------------------------------|-------|-------|-------|---------|---------|----------|-----------|-----------|
| Connect Timeout(sec) | 20 | 15 | 10 | 8 | 5 | 4 | 2 | 1.5 |
| Disconnect Timeout(sec) | 30 | 20 | 14 | 12 | 8 | 6 | 3 | 2 |
| Gossip Timeout(sec) | 4 | 4 | 4 | 4 | 4 | 4 | 4 | 4 |
| Failure Detection Timeout(sec) | 12 | 12 | 12 | 12 | 12 | 12 | 12 | 12 |
| Low Threshold | 5 | 6 | 6 | 6 | 7 | 8 | 9 | 10 |
| High Threshold | 10 | 11 | 11 | 11 | 12 | 13 | 14 | 15 |

Table 3
Parameter values for mobility-adaptive ad-hoc Araneola

Route Driven Gossip: The gossip period T_g is set as follows, with respect to the average node speed, avg_speed .

$$T_g = \begin{cases} 2 \text{ s} & \text{for } avg_speed < 4 \text{ s} \\ 1 \text{ s} & \text{for } avg_speed \geq 4 \end{cases} \quad (4)$$

These values have been empirically determined to improve reliability, while maintaining the control overhead comparable to the one induced by RDA. Decreasing the *gossip timeout* periods would yield higher reliability but always at the cost of increased overhead.

7 Simulation Results

Below we present and discuss our simulation results. Unless noted otherwise, the average node speed varies from 0 to 20 m/s and there is a single multicast source which injects messages with rate kept constant at 1 msg/s. Throughout the discussion we classify mobility as very low (0-2 m/s), low (2-5 m/s), moderate (5-10 m/s) and high (> 10 m/s).

7.1 Route Driven Araneola Variations Evaluation

We perform comparison among various RDA implementations described in Section 5.2 and listed in Table 4. Each of these implementations incorporates different optimizations. This series experiments aims at confirming and quantifying the performance gains from the modifications on the base RDA protocol, thereby allowing us to determine the best set of optimizations for RDA.

7.1.1 RDA Performance Comparison

Due to space limitations, we do not include the detailed performance comparison among RDA variations. The interested reader may refer to [?] for the related experimental results. The comparison is performed in terms of reliability, control overhead and average end-to-end delay. We summarize the results of that series of evaluations in subsection 7.1.3.

| RDA variation | Characteristics |
|----------------|---|
| Baseline(RDA) | Low=5, High=10. Gossip/Connect/Disconnect timeout = 5/15/20 s. |
| RDA-MA | Mobility-adaptive. See table3. |
| RDA-MAP | Promiscuous RDA-MA. DSR passes routed Gossip and Data messages to RDA. |
| RDA-MACR | RDA-MA that connects only to reachable nodes. |
| RDA-MATA | Topology Aware RDA-MA. The active view is weighted according to the hop count to destination. |
| RDA-MASF | RDA-MA with scoped flooding. Floods Route Requests with low TTL (5). |
| RDA-MADRP | RDA-MA with repair data-pull for missing messages |
| Araneola-MANRD | Non-route-driven Araneola-MA over DSR. It does not depend on route state. It is promiscuous and performs repair data-pulls. |

Table 4
RDA variations.

7.1.2 Overlay Properties

Below we discuss the properties of the overlay constructed by the Route Driven Araneola variations. We consider the metrics listed in Section 6.2, which are *diameter*, *average degree* and *average connection duration*.

Diameter Comparing mobility adaptive and topology aware RDA (RDA-MATA) to the non-topology-aware RDA-MA (Fig. 5), we observe that for no and very low mobility RDA-MATA’s overlay diameter is larger. We attribute this result to the non-uniform manner in which neighbor candidates are selected. For low to moderate speeds however, random mobility introduced sufficient randomness in the overlay construction, thereby allowing RDA-MATA to compare favorably to RDA-MA. The reason for the increase of diameter with increased speed is route failures that cause removal of edges in the overlay structure. Since RDA-MATA incurs less route failures than RDA-MA, its overlay diameter is lower. However, the differences between the values for the two variations are lower than the estimated confidence intervals. We do not consider high speeds (above 10 m/s) because low reliability skews diameter-related measurements.

Below we discuss the diameter property of the regular overlay graph that RDA attempts to approximate with respect to theoretical predictions [?]. We base our discussion on the end-to-end and hop-count metrics as they were defined in Section 6.2.

For $|V| = 100$ and $k = 4$, Equation 3.1 yields diameter between 4 and 9.⁵ Therefore, the expected maximum hop count is 4 to 9 and the expected maximum end-to-end delay is $2 \cdot 9 \cdot \text{gossip_timeout}$ (gossip_timeout as defined in Section 3 and set in Table 3).

As can be observed in Fig. 5(b), when mobility is below 2 m/s, the maximum hop count for all variations approximates the predicted diameter (between 5 and 10). However, hop-count increases dramatically with increased mobility (it reaches the maximum 22 for the RDA-MASF variation for speed equal to 9 m/s). Over this speed, hop count decreases but this is attributed to the fact that messages are not successfully relayed along the overlay neighbors. The end-to-end delay results demonstrate poor RDA latency performance.

The maximum end-to-end delay is considerably above the theoretical predictions described in the earlier paragraph. The inconsistency between the end-to-end delay results and the theoretical predictions indicates that for very low mobility, over time, a regular overlay is successfully created, however it is disrupted by temporary failures, which cause delays in the message dissemination. A type of these failures is data and gossip message losses, which result in retransmission of requests at a later time. The second type is temporary removal of connections, as routes between two neighbors cease to exist and re-establish connection later, when they are able to request the missing messages. These failure types are not observed under the network conditions studied in [?]. That analysis does not examine data message losses due to UDP packet losses, while *heard_from* entries are removed upon neighbor removal, preventing neighbors to request missing messages when an overlay connection relationship is re-established.

⁵ Note that this inequality holds a.a.s, thus for $|V|$ small the resulting range is not accurate.

The observed end-to-end-delay, as well as the overlay degree for all variations indicate that, for very low to low mobility overlay creation is successful. For moderate to high mobility the overlay structure deteriorates, as it is derived by the increased end-to-end delays. However, judging by the degree (3-6) shown in graph 6(b), connectivity remains sufficient.

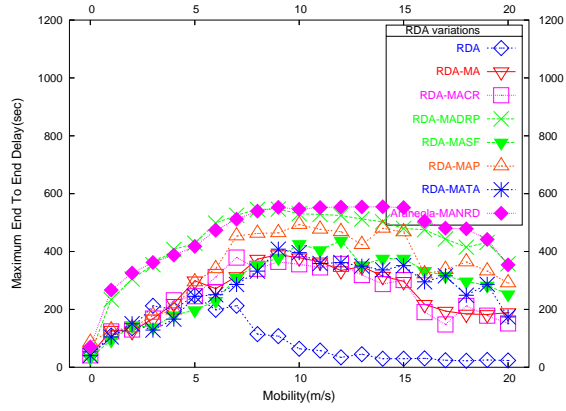
Average Degree RDA-MATA does not yield reduced connectivity, as can be inferred by Fig. 6(b) and by reliability measurements. From Fig. 5, we reach similar conclusions regarding the scoped flooding RDA-MASF variation. RDA-MASF yields higher overlay diameter than RDA-MATA for moderate mobility.

From Fig. 6(b) we determine that baseline RDA is not effective in constructing the desired overlay for other than very low speeds. For RDA mobility-adaptive variations, parameters are tuned to achieve higher degree connectivity. For the values listed in Table 3, overlay degree is maintained in the interval (4,6), for all simulated speeds and traffic load. For example, at 0 m/s the overlay average degree is 5.08 and at 10 m/s it is 5.03. Recall that the target degree (*low* threshold) is set to five. Non-route-driven Araneola (MANRD) is the least effective in maintaining the required degree for higher speeds. This behavior is attributed to the high control packets loss rate by route-discovery-induced contention and broken links that stem from the lack of route awareness. As expected, we observe high correlation of protocol performance with achieved degree. In addition, it is apparent that the strategy of increasing threshold with mobility is effective in approximating the target k -regular overlay.

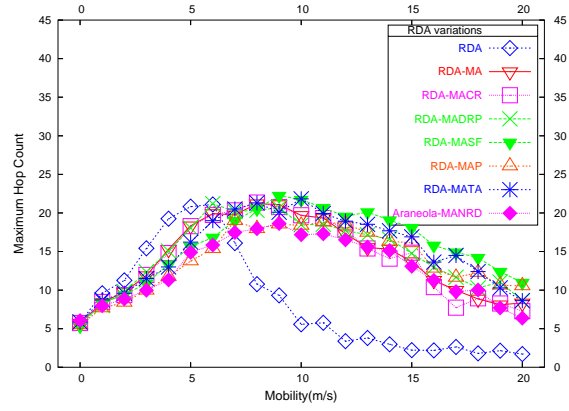
For non-mobility-adaptive RDA under very low and low mobility, we observe that the average degree approximates the low threshold (5). Thus, in a stationary or mildly volatile ad-hoc environment the overlay maintenance task is successful in creating the k -regular overlay. For 5 m/s, the average degree drops to 3. The decrease in degree is due to packet loss during the negotiation of connection among peers. The baseline RDA does not cope with packet loss by increasing the *high* and *low* thresholds, effectively increasing the number of members toward which it dispatches connection requests. At high speeds, the overlay degree drops below four, even for the mobility adaptive variations.

Average Connection Duration Fig. 6(a) shows the average lifetime of overlay connections as a function of mobility and packet injection rate respectively. We observe that for moderate and high mobility the lifetime of connections is substantially reduced, indicating increased overlay instability. For no mobility, connection duration is relatively low (30 s). We believe this is due to poor load balancing. Araneola-MANRD exhibits the lowest connection duration. Since it does not establish connection relationships according to reachability, *gossip* messages are more likely to be lost, resulting to nodes removing their overlay connections.⁶

⁶ We do not consider baseline RDA's connection duration. as a function of mobility due to the very low overlay degree for moderate and high mobility.

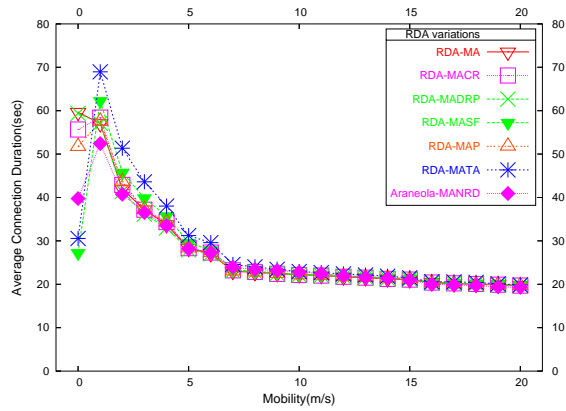


(a)

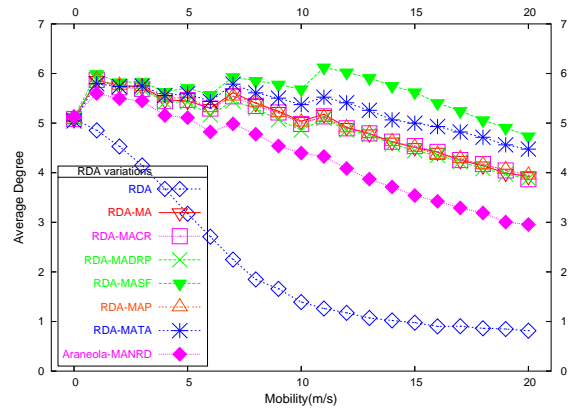


(b)

Fig. 5. Overlay properties of RDA variations as a function of mobility. A single multicast sender injects 1 msg/s. Graph (a) shows *Maximum end-to-end delay* and (b) *Maximum hop count*.



(a)



(b)

Fig. 6. Overlay properties of RDA variations as a function of mobility. A single multicast sender injects 1 msg/s. Graph (a) shows *Average connection duration* and (b) *Average degree*.

7.1.3 RDA Evaluation Conclusions

The aforementioned results and related results in [?] can be summarized in the following conclusions:

- Adaptation to mobility is required for the protocol to cope with frequently changing topology.
- By studying the behavior of RDA-MAP and RDA-MATA variations, we determined that routing layer promiscuousness and topology awareness can improve all four examined aspects of performance.
- Repair data-pulls improve packet delivery ratio.
- It is substantially beneficial to adjust the overlay according to the reachability state, connecting only to reachable members.
- Non-route-driven Araneola (MANRD), which is promiscuous but does not consider topology and reachability, compares unfavorably, in terms of reliability, to mobility-adaptive route driven variations, for moderate and high mobility. Furthermore, non-route-driven Araneola incurs higher signaling, This

is expected, since this variation does not consider route state and hop-proximity, therefore it performs more frequent route discoveries and uses costlier paths.

- For high mobility (above 10 m/s) the reliability results in [?] (0.2-0.4), indicate that increasing the rate overlay control messages are exchanged (Table 3), causes network contention without effectively maintaining a reliable service..
- Network proximity-based selection of neighbors (RDA-MATA, RDA-MASF) does not substantially increase the overlay diameter, although it reduces the randomness in overlay creation.

Our simulation results indicate that the proposed application layer multicast overlay is a viable solution in a low to moderate mobility ad-hoc environment. Although unicast routing imposes significant control overhead, it yields significant connectivity and randomness gains. We present detailed discussion of the unicast routing overhead in Section 7.2.2.

7.2 Protocols Evaluation

This series of experiments compares the communication service of overlay deterministic flooding (ad-hoc Araneola), overlay rumor mongering (RDG) and link-level flooding. This study aims to determine whether overlay deterministic flooding in volatile multi-hop topologies yields the gains over gossip that were experimentally observed in a LAN setting [?] and were theoretically predicted in the case of flooding over k -node and k -link connected Harary communication graphs [?].

The overlay schemes are compared against “brute-force” link-level flooding to determine in which cases epidemic application layer multicast is more suitable than simple link-level broadcast schemes. In [?] we performed in-detail evaluation of probabilistic and controlled link-level flooding variations under the described simulation network settings. We have determined that link-level flooding in which nodes relay unseen packets with probability one (denoted as FL) is the most efficient and reliable among them.

Furthermore, the investigated protocols are evaluated in environments with both multicast and unicast traffic. Throughout these experiments the multicast traffic load remains constant, whereas mobility and the number of unicast senders is variable. These evaluations seek to determine whether the imposed unicast traffic amortizes the cost of application layer multicast, enabling it to compare favorably to link-level flooding. Intuitively, background unicast traffic causes the routing function to update routing tables, reducing the routing signaling induced by the multicast functions. At the same time, the unicast senders are benefited by the routes resolved for multicasting.

7.2.1 Protocols Performance Comparison

Below we compare the reliability, control overhead ⁷ and average end-to-end delay of: (a) RDA that incorporates all the modifications that were determined to be beneficial in Section 7.1 (RDA-OPT);

⁷ Low control overhead has a two-fold impact on reliability: (a) data packets are more reliably transmitted because the probability of collisions is reduced and (b) overlay maintenance packets are more reliably transmitted, inducing a more structured and consistent overlay over which data messages are more effectively disseminated.

(b) RDG; (c) “brute-force” flooding (FL) and (d) mobility-adaptive Araneola over a link-state protocol (Araneola-MALS).

Araneola-MALS (also referred to as Araneola-LS) is topology aware, promiscuous with respect to FSR’s routing function and performs repair data-pulls. We evaluate it to obtain insight on the actual, if any, benefits of the use of route-driven membership and reactive routing.

RDG- F denotes the rumor protocol with fanout F . Since in RDA each host advertises a new message only once, RDG parameter t_g is set to 1. RDG’s v , r and m parameters are set to 2, 1 and 3 respectively (see Section 5.1 for definitions of RDG parameters). The evaluated RDG variation is topology aware and routing layer promiscuous, as described in Sections 5.6 and 5.3(Araneola augmentation (f)), respectively. It also performs scoped flooding of route queries. Topology awareness in Araneola-LS, RDG and RDA is implemented according to Section 5.6 with $p_s = 0.66$.

Recall that for Araneola-LS, RDA and RDG, the maximum number of 512-byte messages per data packet is four. The link-level flooding data packets are sent immediately upon generation, therefore each data packet contains a single 512-byte message.

For fairness of comparison, both ad-hoc Araneola and the investigated variation of RDG are pull-based (see Sections 5.2 and 5.1).

In Araneola, a node with degree low forwards the message digests, downstream only, to $low - 1$ neighbor nodes. In the gossip protocol a node with fanout F forwards the digests to all the randomly selected F nodes. Therefore, the gossip protocol is comparable, in terms of amount of message digest information inserted in the network, to Araneola with $low = F + 1$. Since parameters in ad-hoc Araneola aim at maintaining the overlay degree between five and six, our experiments juxtapose ad-hoc Araneola and RDG with F equal to five and six.

Reliability Fig. 7 depicts reliability, transmission/delivery overhead, and average end-to-end delay as a function of average node speed. As can be seen in Fig. 7(a), RDA-OPT has satisfactory reliability from very low to moderate mobility. However, it degrades by more than 50% for high speeds. RDG-6 has lower reliability than RDA-OPT for speeds below 8 m/s, and RDG-5 is less reliable for all speeds.

As expected, the route-driven variation of Araneola yields reliability gains over link-state Araneola, but this does not hold under high mobility. Although overlay relationships among nodes are established according to reachability, as topology volatility increases, nodes do not remain reachable long enough so costly route discovery and routing errors occur frequently. As a result, Araneola-MALS is the most reliable among the application layer protocols for speeds above 11 m/s.

The above result is in accordance with results in [?], which showed that FSR is more suitable than DSR when mobility is moderate and most links are utilized. FSR periodically exchanges link-state information in a proactive manner, while DSR performs route discoveries on demand. In ad-hoc Araneola, the majority of the links in the network are used to provide paths from multiple senders to multiple receivers, therefore the proactive approach is not inefficient, since the significant portion of the proactively resolved routing information is useful. On the other hand, the on-demand approach is not as plausible

because it requires many flood-based discoveries to support this mode of communication. At the same time, FSR is designed so as to greatly reduce the cost of exchanging link-state information (see Section 4.2). As discussed in more detail below, in this case, the reliability gains of RDA-OPT do not come at the expense of increased control overhead.

The message delivery ratio of “brute-force” link-level flooding (FL) is between 0.9 and 1.0 for all simulated speeds. At the same time, the control overhead is significantly lower than the one incurred by the other dissemination mechanisms. Flooding compares favorably to RDA-OPT, Araneola-LS and RDG for almost all configurations. This stems from the fact that it makes minimal assumptions about the underlying topology and imposes less signaling load.

Control Overhead Both RDG variations inject 2 to 4 times less control bytes to deliver data than RDA with the exception of very low to low speeds (Fig. 7(c)). For low to moderate speeds, RDG-6 induces approximately equal delivery overhead to link-state Araneola’s. At high speeds though, the latter exhibits almost double the overhead. Therefore the higher reliability of Araneola-MALS at high speeds comes at the cost of increased overhead, which is still two to three times less than the overhead induced by RDA.

As can be seen in Fig. 7(b) and 7(c), the control overhead of link-level flooding is 10 to 200 times lower than the ones of Araneola and RDG. This difference is partly due to the direct interaction with the MAC layer and the absence of routing and application layer signaling. This result signifies link-level flooding’s superiority in terms of scalability when all network nodes are receivers. Note however, that the way we calculate the delivery overhead (see Section 6.2) favors flooding.⁸ Nevertheless, even if we included redundant data packets in the control overhead, link-level flooding would still have substantially less overhead than the rest of the protocols. This can be safely inferred by the fact that the actually computed control overhead is substantially smaller and from the much shorter duration of flooding’s simulations.

Average End-to-end Delay In Fig. 7(d) we observe that RDG’s end-to-end delay is less than ad-hoc Araneola for all speeds. Below 3 m/s the RDG-6 latency is roughly equal to the delay RDA-OPT and Araneola-LS yield. This indicates that in the simulated network, under that low and very low speed, the diameter of the approximated by RDA random regular overlay of degree k almost matches the diameter of RDG’s random non-regular overlay with fanout k , which is in not in accordance with theoretical predictions ([?] and [?]). However, these predictions are not accurate for small sized networks and do not consider interaction at layers 2 and 3. The fact that the two end-to-end delays are comparable, is still a strong indication that the random regular geometry is successfully approximated for low mobility. The end-to-end delay exhibited by flooding is the lowest because flooding is not regulated by rounds and

⁸ A great portion of link-level flooding’s overhead consists of redundant data packets, which we do not include in the calculation. This is because duplicate data packets can only be identified at the destination, while the control overhead is derived at the source, destination and the intermediate nodes. This is an issue with the rest of protocols too but it is not as severe; in Araneola and RDG, the sources transmit data only when they are explicitly requested by the destinations.

multicast data are relayed instantly.

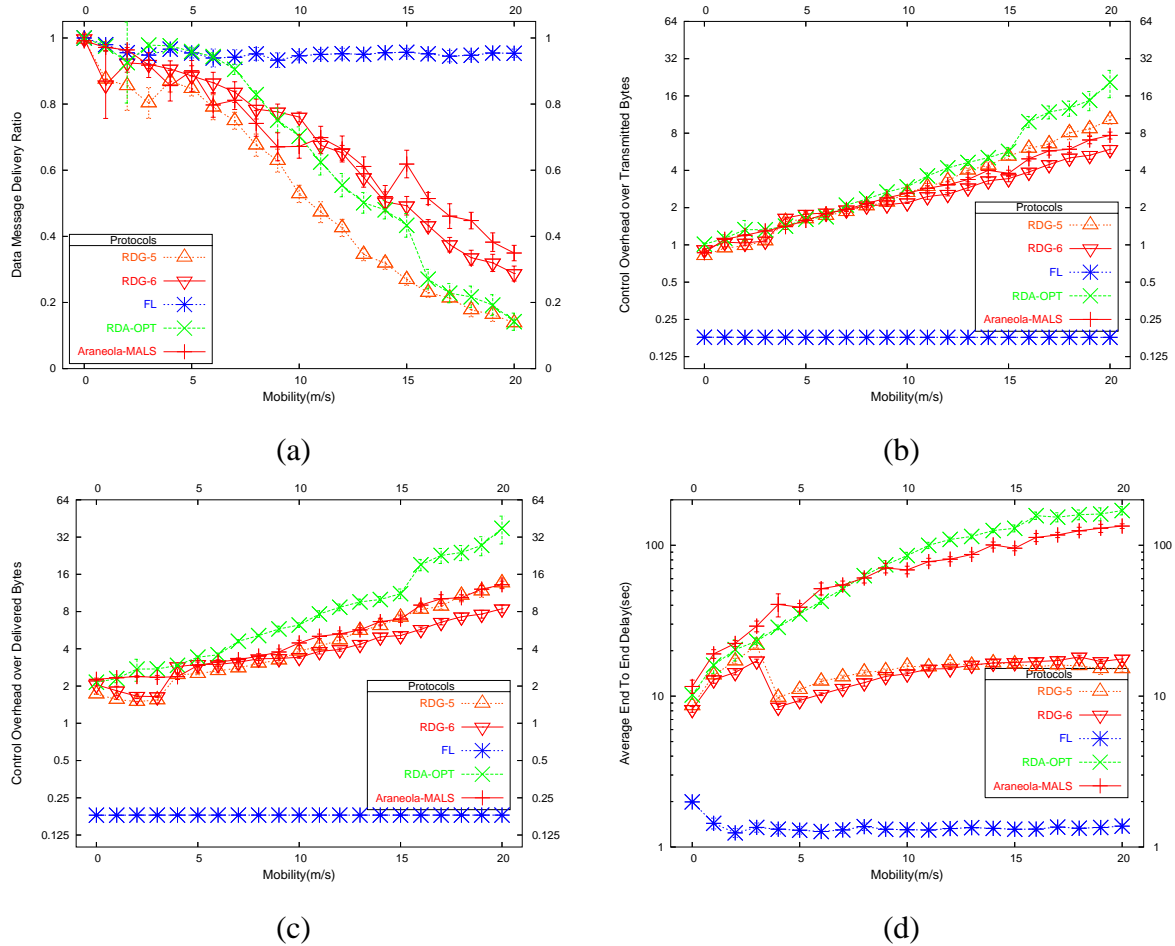


Fig. 7. Performance characteristics of protocols as a function of mobility. A single multicast sender injects 1 msg/s. Graph (a) shows *Reliability*, (b) *Control overhead over transmitted data bytes*, (c) *Control Overhead over delivered bytes* and (d) *Average end-to-end delay*.

7.2.2 Control Overhead Decomposition

Below we discuss the composition of the observed control overhead. We study overhead in terms of two components. The 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 frames transmitted for DSR or FSR messages. The latter includes application layer protocol headers, overlay maintenance packets and 802.11 control packets transmitted for Araneola messages.

Fig. 8(a, b) present the decomposition of control overhead exhibited by RDG-6, RDA-OPT and Araneola-LS, as a function of mobility.

For RDA at low mobility, the dominant component is application signaling. As DSR attempts to tackle frequently changing connectivity, which invalidates cached routes, it initiates more frequent route dis-

coveries to acquire more updated and correct routing information. For example at 1 m/s, the application control overhead to deliver data is 0.81 and the respective value for routing is 0.63. At 5 m/s though, routing signaling is greater (1.52) than application signaling (0.89) and the difference between the two increases with speed.

Hence, as topology volatility increases, routing induces more load on the network than overlay maintenance. In addition, for low speed, the overhead can be significantly reduced with further RDA optimizations, which would reduce overlay maintenance transmissions.

In RDG the routing delivery overhead is less than the application for low mobility but this is reversed for speeds above 5 m/s. The routing overhead does not hinder RDA multicast for low speeds. In this range, the application layer overhead is greater than routing. The latter becomes significant as the topology volatility increases (above 10 m/s). In this case the routing to application overhead ratio indicates that the unicast routing function is the main cause of contention due to increase of route errors and more frequent route discoveries.

The routing layer signaling of RDA is 30%, or more, higher than Araneola-LS's routing signaling, except under very low mobility when RDA expends about 15% less routing overhead to deliver data bytes. In Araneola-LS the routing signaling consistently exceeds application signaling. As discussed above in 7.2.1 FSR is more effective than DSR for Araneola's mode of communication in volatile topologies.

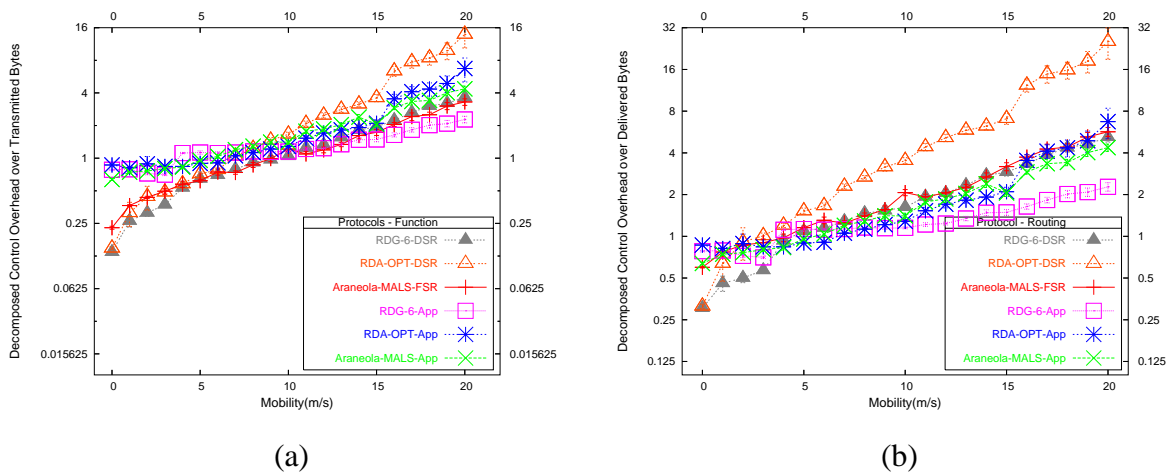


Fig. 8. Decomposition of control overhead of protocols in application and routing signaling as a function of mobility. A single sender injects 1 msg/sec. Graph (a) shows *Decomposed control overhead over transmitted data bytes* and (b) *Decomposed control overhead over delivered data bytes*.

7.2.3 Protocols Evaluation Conclusions

Through the above analysis we reach the following conclusions:

- Link-level flooding is the most reliable and efficient among the examined dissemination methods for the range of simulated speeds, in a small network, in which all nodes are receivers. Unlike the application layer schemes, it is not substantially affected by mobility.

- Optimized RDA, for low to moderate mobility, yields approximately 10% higher reliability than RDG. Similarly RDA performs more reliably than Araneola over link-state in this mobility range. For speeds above 5 m/s, RDA and Araneola-LS with target degree equal to five entail more signaling than RDG with fan-out equal to six.
- For high speeds, Route Driven Gossip with fan-out six degrades more gracefully than the deterministic flooding protocol over reactive routing. We attribute this to the deconstruction of the regular graph structure imposed by the latter, and the increased signaling. RDA is unable to effectively perform deterministic flooding in highly dynamic environments where RDG-6 is a more reliable solution.
- For low to moderate mobility, RDA is more reliable than Araneola-LS, but this does not hold for high speeds. Araneola-LS is the most reliable among the application layer protocols at speeds above 10 m/s. The advantages of reactive routing protocols and of route-driven views cease to exist under highly volatile topologies and when the majority of the links is utilized. Under moderate and high mobility and for the type of network traffic that Araneola injects in the network, reactive source routing consumes more network resources than ad-hoc link state routing.⁹
- The routing overhead does not hinder RDA multicast under low mobility. Below 5 m/s, the dominant overhead component is application layer control messages. Under increased mobility DSR routing imposes higher signaling than application layer signaling, inducing contention.
- The investigated ALM mechanisms are not suitable for real-time communications, since they demonstrate one to two orders of magnitude higher latency and jittering than other multicast mechanisms, including flooding.

7.3 Protocol Comparison under Mixed Traffic

In this section we compare the aforementioned protocols under both multicast and unicast (mixed) traffic. The CRB unicast sources insert messages at a rate equal to 1 msg/s. We present reliability and end-to-end delay for both the CBR unicast (depicted as *Protocol-Unicast*) and the multicast function of the studied multicast protocols (depicted as *Protocol-Multicast*). The control overhead involves control bytes transmitted or delivered by both functions.

Fig. 9 compares unicast and multicast reliability, control overhead and end-to-end delay of RDA-OPT, Araneola-LS, RDG-6 and flooding (FL), as a function of mobility. The number of unicast sources remains constant at 10. Fig. 10 compares reliability, control overhead and end-to-end of the above protocols, as a function of unicast senders. The average node speed is constant at 1 m/s.

Reliability As can be seen in Fig. 9(a), RDA-OPT outperforms RDG-6, in terms of multicast reliability, for speeds up to 15 m/s. Araneola-MALS outperforms RDG-6 for all speeds and it performs better than RDA only for high mobility. Flooding still provides higher multicast reliability than RDG, RDA

⁹ In [?], we also evaluate the performance of the protocols as a function of traffic load. We conclude that the transmission and delivery overhead of RDA-OPT and Araneola-LS is substantially reduced with load. The reason is that, in the case of ad-hoc Araneola, for higher message injection rates, control bytes are used to construct and maintain an overlay structure over which more data messages are efficiently disseminated.

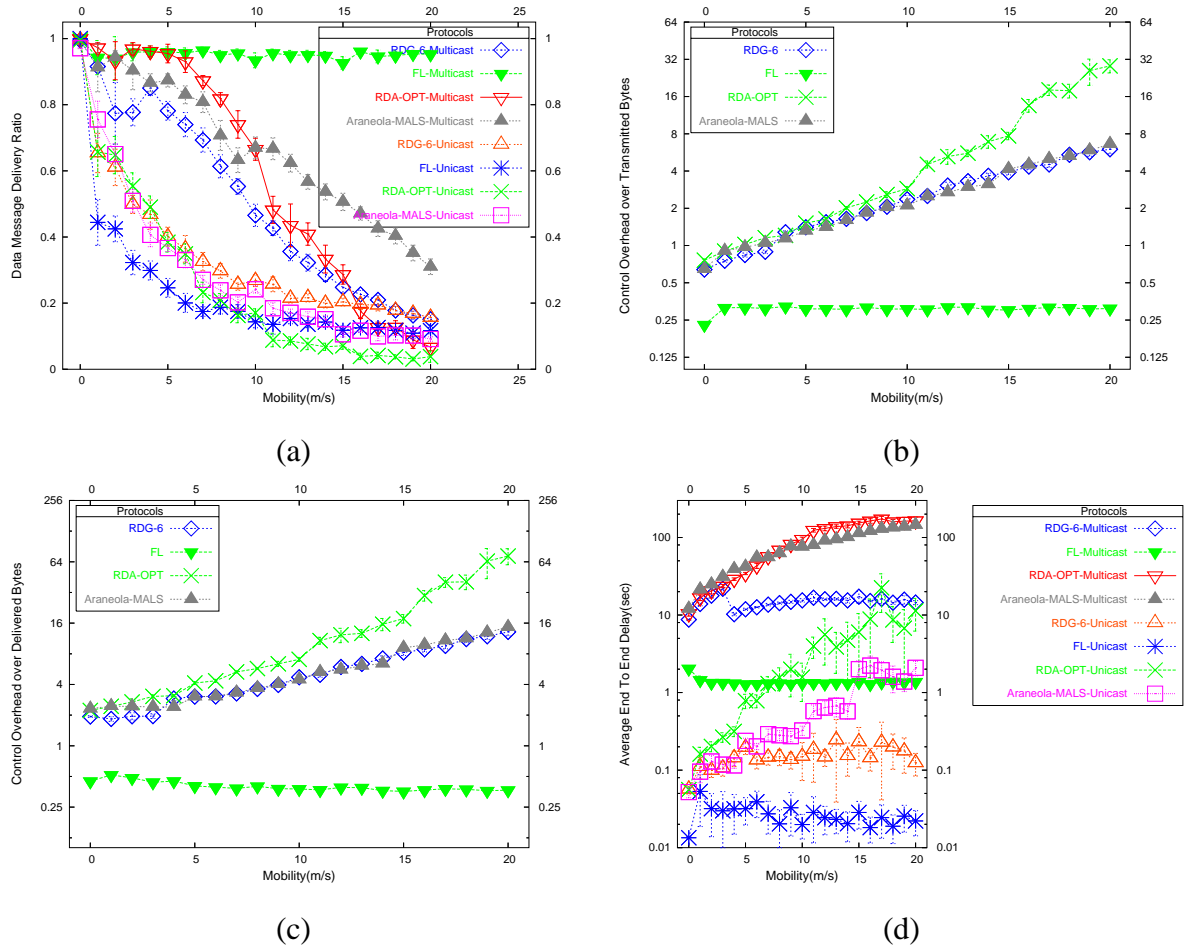


Fig. 9. Performance characteristics of protocols with unicast traffic as a function of mobility. A single multicast sender injects 512 byte messages at the rate of 1 msg/s. 10 unicast senders inject 512 byte packets at the rate of 1 pkt/s. Graph (a) shows *Reliability*, (b) *Control overhead over transmitted data bytes*, (c) *Control overhead over delivered data bytes* and (d) *Average end-to-end delay*.

and Araneola-LS, especially for speeds for moderate or higher speeds.

In Fig. 9(a), we observe that unicast reliability degrades substantially with increased mobility for all schemes. Link-level flooding is not as appealing under mixed traffic. For low to moderate speeds, unicast traffic's reliability under flooding compares unfavorably to the one under ad-hoc Araneola. Link-level flooding does not take advantage of the existing route resolution and interferes with DSR control traffic, causing contention-induced unicast traffic packet losses. RDG-6 is the most resilient among the protocols in terms of unicast reliability.

In Fig. 10(a), we observe that RDG-6 appears very sensitive to the number of unicast senders. This is not the case for the rest of the application-layer schemes. For 10 unicast senders all protocols exhibit roughly 100% reliability. For 50 unicast senders, RDG-6 exhibits very low reliability, while for RDA-OPT and Araneola-LS reliability remains high. In reactive protocols, existing unicast traffic drives route resolution for routes that are requested by the unicast sources. As the number of sources increases, more routes are resolved. The resolved paths are then used by the multicast protocol. Since ad-hoc Araneola

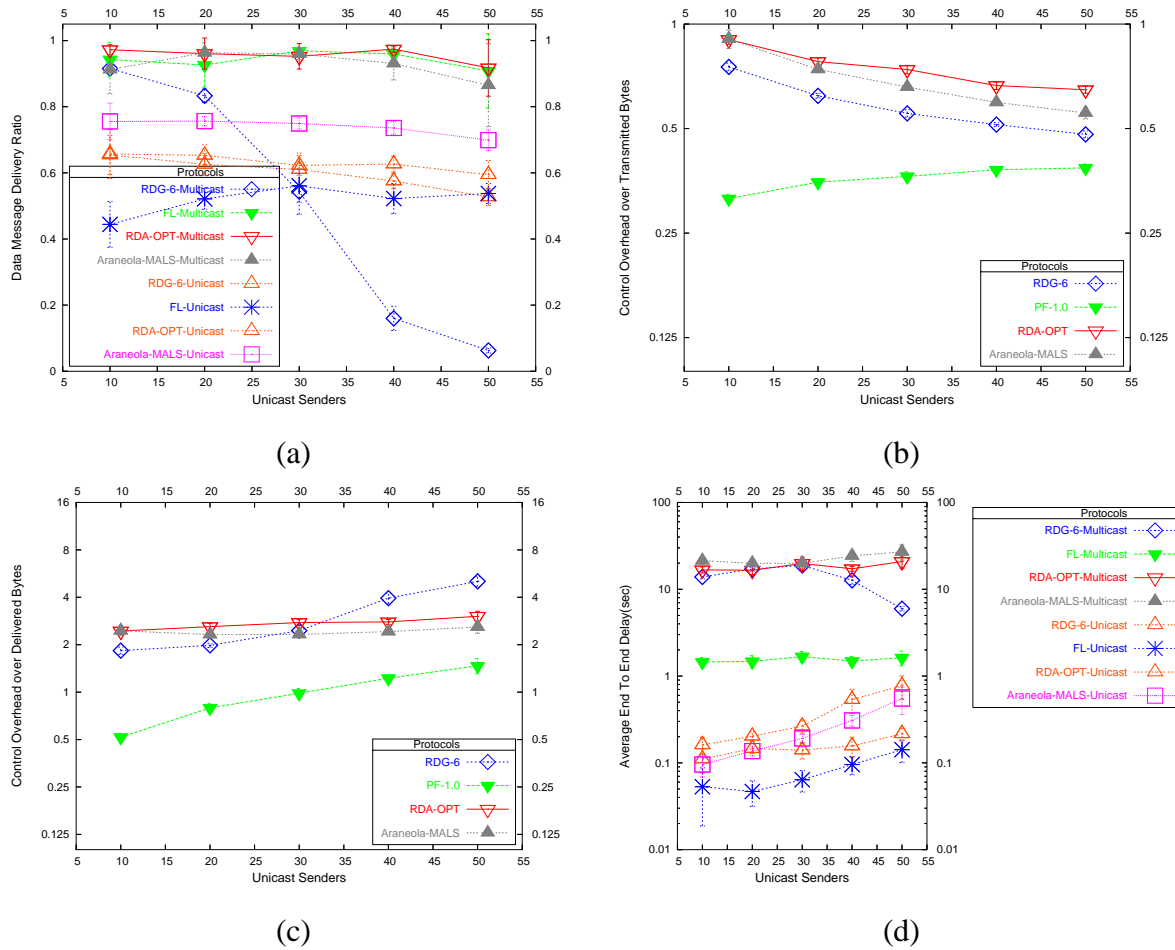


Fig. 10. Performance characteristics of protocols with unicast traffic as a function of unicast senders. A single multicast sender injects 512 byte messages at the rate of 1 msg/s. Unicast senders inject 512 byte packets at the rate of 1 msg/s. The average node speed is 1 m/s. Graph (a) shows *Reliability*, (b) *Control overhead over transmitted data bytes*, (c) *Control overhead over delivered data bytes* and (d) *Average end-to-end delay*.

uses tunneling not only for data transmission but for transmitting overlay maintenance messages, it uses the resolved routes more than RDG.

The above conclusion can also be confirmed using Fig. 8. We see that the routing control overhead for transmitting and delivering data is substantially lower for RDG than it is for ad-hoc Araneola. The difference between the overhead of the two schemes increases with speed. It is apparent that RDA makes more use of DSR. Of course, RDA is not more effective when there is unicast traffic in the background. Rather, its reliability is less affected by the background traffic than RDG's reliability.

The multicast message delivery ratio exhibited by flooding is not affected by the amount of unicast traffic in the network. Note, however, that flooding's (FL) unicast delivery ratio is the lowest among the four schemes. It does not utilize the existing unicast routing function and interferes with flood-based route discoveries.

Control Overhead As can be seen in Fig. 9(b, c), RDA-OPT and Araneola-LS, in the presence of unicast traffic load, induce higher than RDG-6 control overhead for transmitting and delivering data bytes for moderate to high mobility. All three protocols impose substantially higher signaling than flooding, which increases with mobility.

Flooding in the presence of unicast load is more scalable than the rest of the schemes, since it incurs less signaling under all network conditions.

Average End-to-end Delay We examine the multicast and unicast end-to-end delay property of the protocols using Fig. 9(b) and 10(b). At low speeds ad-hoc Araneola exhibits almost equal latency to RDG. Therefore, under mixed traffic too, the diameter of the approximated random k -regular overlay is approximately equal to the one of the non-regular random overlay with fanout k . For more dynamic topologies the regular overlay structure deteriorates and RDG-6 outperforms link-state ad-hoc Araneola, which in turn compares favorably to RDA.

Regarding unicast end-to-end delay, we observe that flooding (FL) incurs the lowest (0.01-0.05 s). This is due to the reduced control overhead, which allows frames to be transmitted with less contention.

From the above, we conclude that in the case of mixed traffic, unicast traffic aids in updating the routing tables and amortizes the overhead that the reactive routing protocols introduce. Hence RDA, which utilizes the underlying tunneling mechanism to a high degree, becomes more reliable than gossip as unicast traffic load increases. We also conclude that the investigated application layer schemes provide higher than brute-force flooding unicast reliability.

8 Discussion and Conclusions

In this work we address the problem of reliable and scalable ad-hoc multicast. We focus in application layer epidemic message dissemination algorithms. Work in [?] and [?] 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. We investigate the viability of this approach in the ad-hoc environment. We propose ways to deploy deterministic flooding in the ad-hoc network and we compare its performance with overlay-based gossip and link-level flooding.

Our study shows, that under certain network configurations, deterministic flooding over a randomly created, expander application layer overlay can be used in mobile multi-hop networks to achieve better performance than overlay-based gossip schemes. The conditions to which the expander-overlay-based protocol must adhere to achieve improved performance are in summary the following:

- The protocol is aware of the underlying connectivity and is parsimonious in consuming bandwidth. RDA achieves this by using a fine-tuned reactive source routing protocol (DSR), being topology-aware, mobility adaptive, route-driven and promiscuous with respect to the routing function. Araneola

over FSR achieves this by utilizing a proactive protocol properly adapted to reduce the amount of link-state information. Furthermore, link-state Araneola is topology aware and promiscuous to the IP layer.

- The network mobility and churn is not high. The regular overlay structure deteriorates dramatically for speeds over 10 m/s.

The investigated overlay-based protocols become reliable and efficient through a cross-layer design. Our experimental results show that in the ad-hoc network, the application layer cannot afford being oblivious to the lower layers. The underlying protocols cannot provide reliability guarantees at a low cost. Therefore, the application layer needs to adapt to the state of the underlying layers and use it to make informed connectivity decisions.

By comparing epidemic overlays, such as Route Driven Araneola and Route Driven Gossip with simple link-layer flooding, we ascertain that link-level flooding is the most efficient and reliable among them, under the examined ad-hoc network configurations and when all nodes act as receivers. These results are in accordance with work in [?] and [?] and extend conclusions to comparisons with the random overlay approach. Flood-based schemes incur signaling overhead solely by duplicate data packets, whereas the investigated probabilistic application layer schemes incur relatively high routing cost. Yet, epidemic application layer schemes exhibit satisfactory reliability under low to moderate mobility. However, this comes at the expense of scalability, especially in highly dynamic environment. The performance of ad-hoc Araneola degrades substantially in moderate to high mobility, when the overlay structure deteriorates.

The comparison of the protocols in networks with both multicast and unicast traffic reveals the improved practicality of probabilistic ALM in these environments. Unicast traffic provides route resolution which is used by the multicast functions, preventing costly route discoveries. In a similar way, unicast traffic utilizes paths resolved for multicast, leveraging the utility of existing route state. In the same context, unicast reliability under flooding is low because flooding does not utilize existing route state. On the contrary, it interferes with flood-based route discoveries.

9 Future Work

In realistic ad-hoc network deployment scenarios (e.g. disaster recovery or battlefield), the network consists of a great number of collaborating devices (sensors, relay nodes etc) possibly deployed over wide areas. Therefore, scalability needs to be addressed rigorously. We expect large scale simulations (500-1000 nodes) to reveal many scalability-related aspects of the protocols, such as optimal partial membership view size.

We are considering optimizations to render the overlay more resilient to mobility. One optimization strategy involves increasing the degree of path redundancy by correlating the target degree with the locally observed connectivity. A simple implementation sets the target degree inversely proportional to the number of one-hop gossip packets that a node overhears promiscuously. Optionally, we could apply the same technique to determine the degree of message redundancy. This technique is reminiscent of [?], which describes an architecture where gossip occurs with highest probability toward nodes with lower

degree. In [?], the authors make a similar suggestion proposing a bimodal scheme that retransmits with higher probability if a neighbor's degree is below the specified threshold.

10 Acknowledgments

We are grateful to Keith Marzullo and Geoffrey Voelker, for providing research directions through many stimulating discussions. Without their advice and support this work would not have been possible.