A Comparison of Group-based Data Persistence Techniques in MANETs

Aaron M. Rosenfeld, Robert N. Lass,
Dustin S. Ingram, and William C. Regli
Department of Computer Science
Drexel University, Philadelphia, PA
Email: {ar374,urlass,dustin,regli}@cs.drexel.edu

Joseph P. Macker
Information Technology Division
Naval Research Lab, Washington, D.C.
Email: joseph.macker@nrl.navy.mil

***Abstract*—Maintaining a consistent set of state information across applications in tactical edge, mobile ad-hoc networks (MANETs) is a challenging, yet mission-critical task. This paper analyzes the performance and effectiveness of eight approaches to data persistence with three different persistence requirements. This is done by introducing COPE, a middleware framework for reliable data delivery that allows different protocols to be configured at runtime. Our empirical results show that Trickle combined with either classical flooding (CF) or NORM provides the best combination of effectiveness and low overhead, compared to other approaches.**

I. Introduction

Maintaining a consistent set of state information across group-oriented applications in tactical edge, mobile ad-hoc networks (MANETs) is a challenging, yet often mission-critical task. Unlike in enterprise networks, mobile applications requiring long-term information consistency among a group of hosts cannot always use classical reliable transport or rely heavily on a centralized server due to network partitioning, high rates of packet loss, and inconsistent network views.

For example, messaging applications often need to continue timely delivery of certain message information even after periods of outage or topological disconnection. Situational awareness systems must still maintain a resilient common operating picture across the network, even if important quiescent data objects are initially sent during periods of network partitioning or failure.

This challenge has been approached in a variety of ways, but can be divided into two general categories: reliable transport protocols and data persistence algorithms. Reliable transport protocols typically resides at the Transport Layer of the TCP/IP model, and provide specific stream or session-based reliability to higher layers. Data persistence algorithms, residing between these protocols and the Application Layer, focus on delivering data even if a host is disconnected or out of communication range during its first transmission.

Some existing work, such as delay-tolerant networking (DTN) [1], employ a cross-layer approach, combining data persistence algorithms and transport level reliability. However, only recently has the interaction of short-term transport-layer reliability and long-term application-layer data persistence begun to be studied [2].

This paper focuses on examining these interactions in the context of group-based data persistence. In this domain, mobile network nodes are involved in similar activities and are therefore interested in much of the same information. Further, multiple nodes may be involved in the transmission, storage, and forwarding of a single piece of data.

The tradeoffs of different group-based data persistence methods are investigated by first comparing a set of candidate protocols independently as a baseline. Then, different protocols are combined and compared in terms of *performance* (the network overhead of the protocol), *latency* (how quickly data is received), and *effectiveness* (how well data is persisted).

Defining how well messages are persisted is dependent upon the class of data. For example, Position Location Indicators (PLIs), which may change frequently due to motion and update reports, may have a very different persistence need than messages which change less infrequently, such as operational orders or quiescent information reports. Therefore, each set of experimental data is analyzed in three contexts, where the data is classified as:

- *Recoverable:* When missed, the data should be recovered but generally does not cause an immediate error (e.g. chat messages)
- *Replaceable:* When missed, the data will be replaced by a more recent version (e.g. GPS location)
- *Aggregate:* When missed, the data must be recovered before the application can process any later versions of the message (e.g. sequential `diffs` of a file)

II. Background

Although tactical MANETs present many networking challenges, most applications running in these environments still require data to be delivered in a reliable manner. To address this requirement, two approaches are typically used: reliable transport protocols and data persistence algorithms.

A. Reliable Transport Protocols

Many issues which arise in mobile networks can be solved with short-term, transport-level reliability. For example, simply resending a message can resolve packet errors due to temporal interference and short term channel errors.

Network transport protocols for large-scale group dissemination have been developed since the early 1990s. The Scalable Reliable Multicast (SRM) protocol [3] was developed to provided localized, many-to-many content repair for receivers with missing data. The Multicast Dissemination Protocol (MDP) [4] extended classical UDP multicast using a one-to-many model to provide a scalable, negative-acknowledgment (NACK) protocol with missing message discovery, repair, and retransmission. MDP was one of the first protocols to use Reed-Solomon (RS) codes, potentially improving scalability and delay with respect to the repair process [5].

MDP's successor protocol, NORM (NACK-Oriented Reliable Multicast) [6], provides similar features but also streaming capabilities, and enables reliable unicast transport as a wireless alternative to TCP. NORM has been well-documented and is transitioning into a standards track Internet RFC [6].

Other related transport work has focused on modifying traditional transport protocols for MANETs. For example, ATCP (TCP for Ad-hoc networks) [7] provides additional features to TCP, improving its performance in networks with high bit-error rate.

Due to their origins in high-fidelity enterprise networks, however, these protocols tend to have limitations when applied to networks with high mobility and topological churn, such as tactical-edge MANETs. Additionally, most of these protocols focus on point-to-point communications, and do not sufficiently address the focus of this paper — group communications.

B. Persistence Protocols

Long-term communication challenges cannot always be solved with reliable transport protocols alone, because they typically focus on short-term network transport sessions. For example, a situational awareness application would need all position information to be repopulated after significant session disruption, system failure, or network failure. In the case of chat clients, a message may need to be delivered even if the intended client was unreachable for a long period of time after the original transmission.

This necessitates a mechanism by which two or more nodes can exchange and synchronize application data, long after the data is originally sent. To do so, messages must be maintained and synchronized across the network.

Early work in this area can be traced to Demers' seminal paper [8] which applied epidemic protocols to distributed database synchronization. Multicast-based consistency techniques where also examined in Van Hook's work on consistency objects in large-scale distributed simulation projects [9].

Since then, many protocols have been developed for the purpose of synchronizing application data. Trickle [10], for example, uses small metadata advertisements to exchange necessary data for synchronization. DIP [11] improved the advertisement process, using a binary-search to reduce overhead. GoSyP [12] employs a more stateful approach, sending unicast messages between pairs synchronizing nodes. Scuttlebutt [13] uses sequence numbers and unicast exchanges to transmit only what is necessary to synchronize nodes.

Although each protocol uses different mechanisms, the overall goal is the same: exchange small amounts of information, locate missing or outdated messages, and then reconcile those discrepancies through message repairing.

III. Motivation

With the vast number of protocol combinations available to application developers, it is often a daunting task to select a proper delivery mechanism for network-bound data. Moreover, this mechanism may need to change based on the deployment scenario.

This challenge arises so frequently, that it is common to integrate reliable transport protocols and long-term persistence algorithms directly into applications. This increases applications complexity, redundancy, and coupling while decreasing adaptability and maintainability.

COPE (Common Object Persistence Engine) attempts to alleviate these burdens, by providing a unified middleware for reliable data delivery and persistence. Instead of the classical approach, where a delivery mechanism is specified at runtime, applications simply pass data to COPE along with desired persistence levels. COPE then locally stores the data, locates other (local or remote) applications that are interested in the data, selects the appropriate delivery technique(s), and persists the data.

However, before the protocol selection process can occur, the trade-offs between different classes of protocols, and the situations in which one should be used over another must be understood. This paper begins to investigate this topic in order to increase the adaptivity of COPE, especially in challenged environments.

IV. The COPE Middleware

COPE is a generic middleware providing reliable delivery and application-layer data persistence. It uses a pluggable architecture allowing various protocols, content-stores, and interprocess communication methods to be specified at runtime. COPE is entirely written in Java, however it is currently being ported to C, to reduce resource utilization and increase portability.

At present, the middleware runs as a daemon which local applications interact with via a TCP socket on the loopback device. Data to be persisted is sent to COPE by connected applications, and recovered data is pushed back to them as it is encountered.

The middleware's primary components are shown in Figure 1:

- *Providers:* Allow bidirectional conversion between native COPE objects and formats which can be sent over the loopback socket. Since these messages remains local, easy to use but verbose formats such as XML may be used.
- *Client Manager:* Handles all local client session sockets and routes information to and from local clients.
- *Content Store:* Provides storage for messages which pass through the COPE instance. This may be volatile (in-memory hash table) or non-volatile (SQLite[1] database).

[1] www.sqlite.org

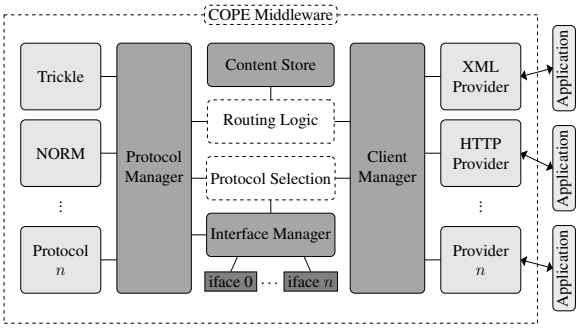

Fig. 1: A high-level view of the COPE middleware.

- *Protocol Manager:* Aggregates all available protocols which could be selected to deliver/receive persistent data objects.
- *Interface Manager:* Maintains instantiated protocols running on network interfaces. Also maintains a list of remote COPE instances on each interface.
- *Routing Logic:* Based on input from the Interface Manager, selects the proper interfaces to which a specific data object should be routed.
- *Protocol Selection:* Selects adequate protocol(s) to deliver a message over a specified interface.

When an application establishes a connection to COPE, it specifies a message format which is available via a Provider (e.g. XML or HTTP in Figure 1). Additionally, it provides a set of subscriptions which are used by the client manager to route recovered data to the proper connected applications.

When a connected application produces data to be persisted, it sends the data along with an application-unique ID, version number, persistence level, and optional custom metadata to COPE. Currently, a message may only be modified (increasing the version number) by the application that originally created it.

The COPE Client Manager receives the data, decoded by the proper provider, and routes it to other locally connected clients if necessary. It is then stored in the content store. After this, COPE queries the Interface Manager to determine over which interfaces the data must be routed. For each one of these interfaces, depending upon the type of attached network (e.g. ad-hoc, access point, wired) and the data persistence level specified by the application, a proper protocol is selected and started.

Currently the selection of a protocol based on these two parameters is defined when COPE is started. For example, one may specify that if the interface is in an ad-hoc mode and the data *must* be delivered, Trickle should be used.

While this works for basic network scenarios, in the future, auto-configuration techniques could be be integrated for heterogeneous networks or scenarios where nodes have multiple operational profiles. Additionally, the process could be dynamic and change over the course of a scenario, based on the network state. This motivates the need to experiment with various network configurations and protocol classes to allow for more future adaptive, automated system designs.

Parameter	Value
WLAN Model	Basic Range
WLAN Range	200m
WLAN Bandwidth	54 Mb/s
Avg. WLAN Delay	50ms

TABLE I: CORE configuration parameters used for all experiments.

Application Layer

Protocol Type	Implementation Used
Polite Gossip	Trickle
Naïve Gossip	Rumor Mongering

Transport Layer

Protocol Type	Implementation Used
Link-local Broadcast	UDP
Best Effort Multicast	Classical Flooding
Reliable Multicast	NORM

TABLE II: Various protocols tested independently and in combination.

V. Experimental Setup

Experiments were run in the Common Open Research Emulator (CORE)[2] using the configuration in Table I. CORE allows unmodified applications to run in real time on virtual network stacks.

Each experiment measured the performance of a single protocol class, or a combination of two protocol classes: one from the application layer and one from the transport layer as listed in Table II. Each emulated node ran a message generator which created a message every five seconds for indefinite persistence and also ran an instance of COPE with the proper protocols.

A. Protocols

For each class of protocol in Table II, a well established implementation was selected for experimentation. Trickle was selected for *polite gossip*, which repeatedly advertises the messages which it is persisting, but remains quiet if its neighbors are advertising the same messages. This results in low-overhead, fast reconciliation of differences.

A basic rumor mongering protocol was selected for *naïve gossip*, which repeatedly broadcasts a list of all messages it is persisting. Based on this, other instances may request or send messages to reconcile differences.

For these protocols, three transport layers were compared. As a baseline, basic UDP broadcast with no forwarding was used. Then, classical flooding (CF) [14] was used wherein every node forwards messages it receives exactly once. The NRL SMF[3] implementation was configured in a mode to provide classical flooding of basic UDP multicast in these experiments. Finally, NORM was used to provide reliable end-to-end delivery of messages. NORM can be configured to operate in multiple modes and the configuration used in this paper is shown in Table III.

[2]http://cs.itd.nrl.navy.mil/work/core/
[3]http://cs.itd.nrl.navy.mil/work/smf

Parameter	Value
Sender Buffer Size	1024 kilobytes
Receiver Buffer Size	1024 kilobytes
Segment Size	1400 bytes
Block Size	64
Parity Size	16
TTL	32
Robust Factor	-1 (Indefinite)
Probing Mode	None
Data Type	NORM Object

TABLE III: NORM configuration for all experiments.

Parameter	Value(s)
Number of Nodes	20
Number of Groups	2, 4, 5
Reference Point Separation	800, 250, 150 meters
Node Separation	25 meters
Max Area	1000×1000 meters
Total Time	5 minutes
Speed	20 ± 5 meters/second
Pause Time	2 ± 2 seconds

TABLE IV: The parameters used for the RPGM mobility model.

As the focus of this paper is on comparing reliable delivery methods at the transport and application layers, basic models of lower level protocols (e.g. MAC and PHY layers) were used. Future work will examine richer lower layer models and interactions.

B. Mobility

All experiments used the Reference Point Group Mobility (RPGM) [15] model with the parameters in Table IV. Nodes were divided into two, four, or five groups and the network connectivity was varied between high and low. The connectivity primarily affected the frequency with which groups came into contact, but also slightly changed the intra-group connectedness.

VI. Results

For all experiments, the performance and latency of each protocol was measured in the same manner. Performance is presented as the number of packets at the IP layer, as measured with `tcpdump`, and latency is the average amount of time between when a message is transmitted and received.

The effectiveness is measured in three different contexts, as introduced in Section I, providing insight into the persistence capability of each protocol class with respect to different message types.

A. Preliminary Analysis

This section analyzes the combination of persistence protocols with NORM or CF under the assumption that the data is of the *recoverable* type, such as chat messages, as introduced in Section I. Specifically, message delivery percentage, as shown in Figures 2 and 3, is used as the metric for protocol effectiveness.

Figure 6 shows that in all cases, when CF or NORM deliver a message, it is done with relatively low delay. Additionally, when messages sent with CF are delivered, they require the transmission of very few IP packets (Figures 4 and 5). Even in high-connectivity networks, though, these approaches failed

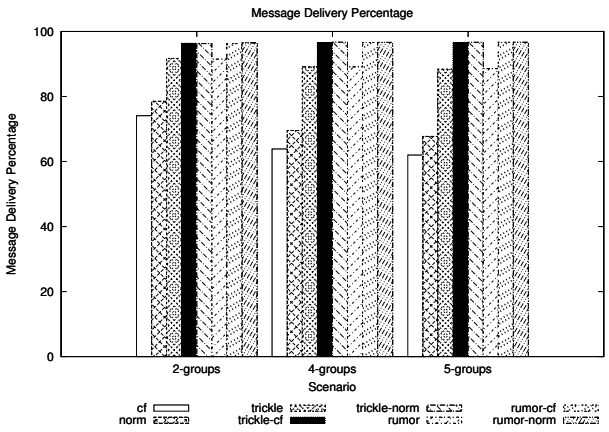

Fig. 2: Message delivery percentages in high connectivity scenarios. Note that the Trickle and rumor mongering variants both provide about the same performance.

to deliver up to 20% of messages, and in low-connectivity networks, up to 90% due to the amount of disruptions and disconnection events that occurred. Figure 7 shows that the standard deviation decreases with higher connectivity, and is low overall.

This provides a clear motivation to use additional persistence techniques in many situations, but the tradeoffs between each of these techniques is not as definitive. Although the rumor mongering variants and the Trickle variants provide about the same performance in the high-connectivity scenario, the rumor mongering variants send up to twice as many IP packets. However, in the low-connectivity scenario, rumor mongering only sends slightly more packets, but performs much better. Trickle's use of polite gossip generally delivers messages more slowly than rumor mongering which constantly broadcasts a list of all local messages. The latter also explains the larger number of packets sent by rumor mongering.

When looking at combination protocols, adding both CF and NORM cause the message delivery percentages of Trickle and rumor mongering to increase. Although NORM delivers slightly more messages, it also sends slightly more packets, making it and CF nearly equivalent in the context of these scenarios.

Trickle demonstrates particularly interesting interactions with the multihop transport protocols NORM and CF. Even though all packets are retransmitted by all nodes, Trickle sends fewer packets overall to deliver a message while also providing higher delivery percentages. This is likely because whenever a Trickle node broadcasts a message, it stifles all other nodes from rebroadcasting for a short period.

B. Replaceable

This section analyzes the effectiveness of each protocol at delivering replaceable messages. There are many examples of replaceable data on the battlefield such as asset GPS positions, which may change frequently.

Effectiveness for this type of data is measured as *staleness*: the duration of time old data remains on each node. It

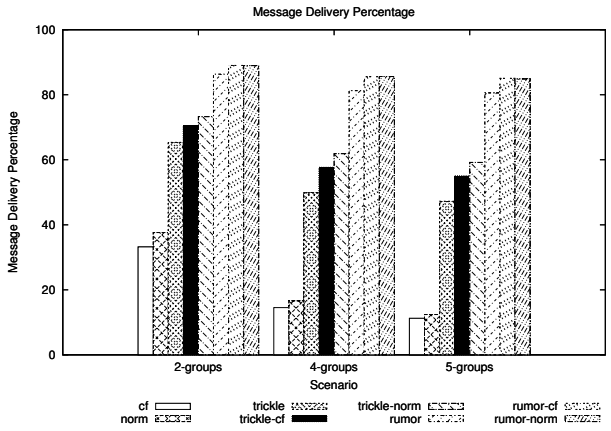

Fig. 3: Message delivery percentages in low connectivity scenarios. Here, the rumor mongering variants outperform the Trickle variants.

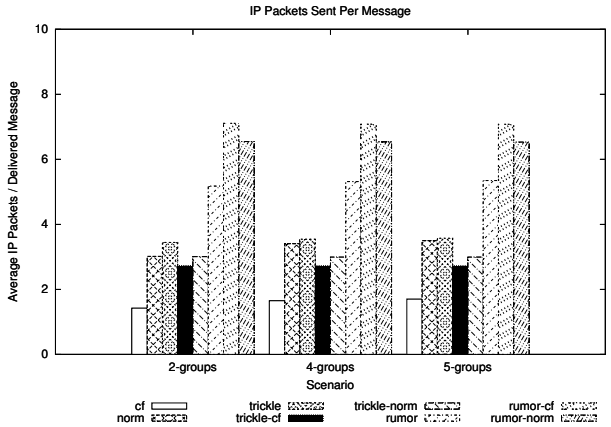

Fig. 4: Total messages sent in the high connectivity scenario.

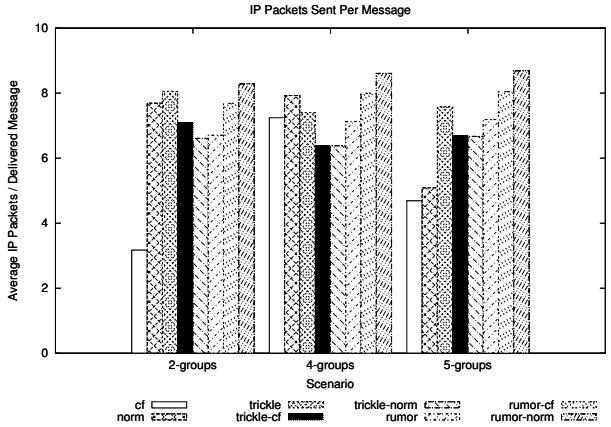

Fig. 5: Total messages sent in the low connectivity scenario.

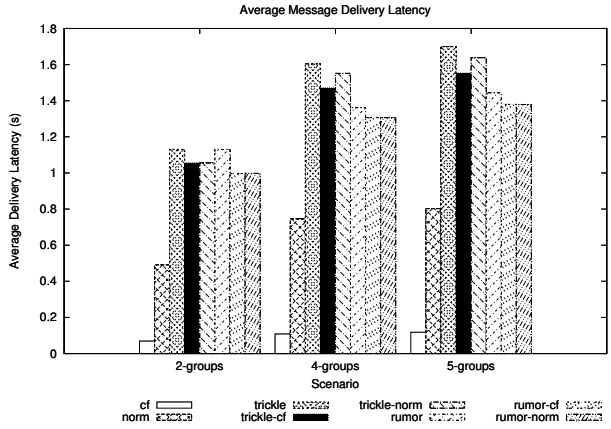

Fig. 6: Average message delivery latency in the high connectivity scenario. The trends are the same for the low connectivity scenarios, but they are not shown due to space limitations.

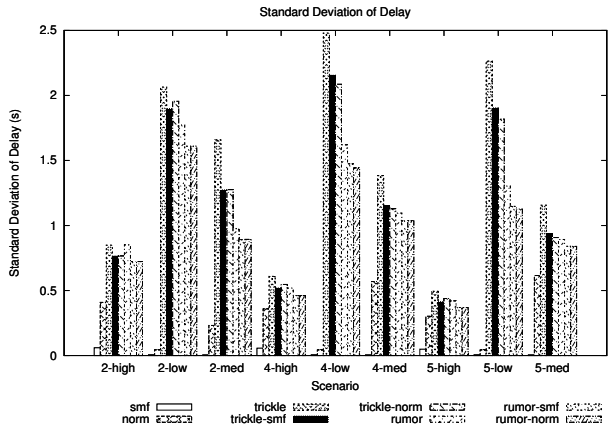

Fig. 7: Standard deviation of the delay.

is assumed that every node maintains a single object, and publishes a stream of updates.

The average message staleness is shown in Table V. Only networks with low connectivity are shown, as all protocols in other experiments showed similar trends.

In each experiment, rumor mongering with either CF or NORM had the lowest message staleness.

C. Aggregate

Aggregate data is a sequence of messages which are combined to form a final application-level message. In tactical MANETs this could be considered a sequence of `diffs` for

Protocol(s)	2-High	4-High	5-High
CF	0.782	1.794	2.326
NORM	0.689	1.568	2.100
Trickle	0.403	0.530	0.558
Trickle & CF	0.373	0.456	0.479
Trickle & NORM	0.359	0.425	0.445
Rumor	0.303	0.323	0.326
Rumor & CF	0.294	0.306	0.308
Rumor & NORM	0.294	0.306	0.309

TABLE V: Average message staleness.

Protocol(s)	2-High	4-High	5-High
CF	8.959	9.436	8.723
NORM	6.380	6.257	6.888
Trickle	5.683	11.210	12.634
Trickle & CF	5.787	10.325	11.545
Trickle & NORM	5.547	9.660	10.691
Rumor	3.605	5.367	5.614
Rumor & CF	2.986	4.169	4.346
Rumor & NORM	2.987	4.136	4.417

TABLE VI: Average previous messages missing.

code updates, or incremental changes to operational orders.

Table VI shows, on receiving a message, how many previous messages must be recovered before producing the combined result. For example, if a node receives code `diffs` for versions 1, 2, and 5, it must wait for 3 and 4 before upgrading from version 2 to 5.

It is assumed that each node maintains one application object and it sends incremental updates as it changes.

Rumor mongering must recover fewer messages on average than the other approaches, with Trickle with CF or NORM requiring slightly more.

VII. Conclusion and Future Work

This paper compared combinations of transport layer protocols and persistence techniques applied separately and in tandem under various scenarios. It investigated their overhead and latency tradeoffs and analyzed the data in different contexts.

Overall, CF and NORM work reasonably well when data needs only a low level of persistence; in highly-connected networks they have been shown to deliver up to 80% of messages.

However, as network connectivity decreases, Trickle and rumor mongering are more effective at persisting all three types of data. Rumor mongering delivers all three types of data to the majority of nodes and does so quickly. It maintained the highest delivery ratio in especially in low-connectivity networks when compared to Trickle and more naïve approaches. However, it also sends significantly more data than these other methods.

Trickle with either CF or NORM appear to be the best combination of delivery percentage, latency, and overhead. Although delivering data slightly slower than rumor mongering, its data usage is quite low. Furthermore, it performed better than CF, NORM, and Trickle alone in all three categories of data.

In the future, several related areas need more detailed examination. First, the operation of these protocols under varying wireless contention and congestion conditions should be further examined since group-oriented network communication can be rather aggressive in regards to redundant messaging. As an example, previous work has shown that the effectiveness of techniques such as CF quickly diminish as the network becomes congested with additional transmissions. The tradeoffs here between overhead, delay, and persistence are non-obvious and need further understanding to be relevant to particular deployment in more bandwidth constrained environments.

Data persistence approaches should also be investigated in more heterogeneous environments, where interface devices may have drastically different behaviors and capabilities. In particular, the performance of protocols hierarchical, unicast environments should be investigated to supplement the broadcast methods presented in this paper.

References

[1] V. Cerf, S. Burleigh, A. Hooke, L. Torgerson, R. Durst, K. Scott, K. Fall, and H. Weiss, "Delay-Tolerant Networking Architecture," RFC 4838 (Informational), Internet Engineering Task Force, Apr. 2007. [Online]. Available: http://www.ietf.org/rfc/rfc4838.txt

[2] K. Scott, T. Refaei, N. Trivedi, J. Trinh, and J. Macker, "Robust communications for disconnected, intermittent, low-bandwidth (dil) environments," in *MILITARY COMMUNICATIONS CONFERENCE, 2011 - MILCOM 2011*, nov. 2011, pp. 1009 –1014.

[3] S. Floyd, V. Jacobson, C. Liu, S. McCanne, and L. Zhang, "A reliable multicast framework for light-weight sessions and application level framing," *IEEE/ACM Transactions on Networking (TON)*, vol. 5, no. 6, pp. 784–803, 1997.

[4] J. Macker and R. Adamson, "The multicast dissemination protocol (mdp)," Internet Engineering Task Force, 1999. [Online]. Available: http://cs.itd.nrl.navy.mil/pubs/docs/draft-macker-rmt-mdp-00.txt

[5] J. Macker, "Reliable multicast transport and integrated erasure-based forward error correction," in *MILCOM 97 Proceedings*, vol. 2. IEEE, 1997, pp. 973–977.

[6] B. Adamson, C. Bormann, M. Handley, and J. Macker, "NACK-Oriented Reliable Multicast (NORM) Transport Protocol," RFC 5740 (Proposed Standard), Internet Engineering Task Force, Nov. 2009. [Online]. Available: http://www.ietf.org/rfc/rfc5740.txt

[7] J. Liu and S. Singh, "Atcp: Tcp for mobile ad hoc networks," *Selected Areas in Communications, IEEE Journal on*, vol. 19, no. 7, pp. 1300 –1315, jul 2001.

[8] A. Demers, D. Greene, C. Hauser, W. Irish, J. Larson, S. Shenker, H. Sturgis, D. Swinehart, and D. Terry, "Epidemic algorithms for replicated database maintenance," in *Proceedings of the sixth annual ACM Symposium on Principles of distributed computing*, ser. PODC '87. New York, NY, USA: ACM, 1987, pp. 1–12. [Online]. Available: http://doi.acm.org/10.1145/41840.41841

[9] D. Van Hook, J. Calvin, and J. Smith, "Data consistency mechanisms to support distributed simulation," in *Proceedings of the 12th Workshop on Standards for the Interoperability of Distributed simulations (Orlando, FL, Mar.)*. Citeseer, 1995.

[10] P. Levis, N. Patel, D. Culler, and S. Shenker, "Trickle: a self-regulating algorithm for code propagation and maintenance in wireless sensor networks," in *Proceedings of the 1st conference on Symposium on Networked Systems Design and Implementation - Volume 1*, ser. NSDI'04. Berkeley, CA, USA: USENIX Association, 2004, pp. 2–2.

[11] K. Lin and P. Levis, "Data discovery and dissemination with dip," in *Information Processing in Sensor Networks, 2008. IPSN '08. International Conference on*, april 2008, pp. 433 –444.

[12] A. Rosenfeld, D. Kusic, and W. Regli, "A gossip-based synchronization protocol for state consistency in distributed applications," *MobiHoc Tactical MANET Workshop*, 2011.

[13] R. van Renesse, D. Dumitriu, V. Gough, and C. Thomas, "Efficient reconciliation and flow control for anti-entropy protocols," in *Proceedings of the 2nd Workshop on Large-Scale Distributed Systems and Middleware*, ser. LADIS '08. New York, NY, USA: ACM, 2008, pp. 6:1–6:7. [Online]. Available: http://doi.acm.org/10.1145/1529974.1529983

[14] J. Macker, J. Dean, and W. Chao, "Simplified multicast forwarding in mobile ad hoc networks," in *MILCOM 2004 Proceedings*. IEEE, November 2004.

[15] T. Camp, J. Boleng, and V. Davies, "A survey of mobility models for ad hoc network research," *Wireless Communications & Mobile Computing (WCMC): Special Issue on Mobile Ad Hoc Networking: Research, Trends and Applications*, vol. 2, pp. 483–502, 2002.