The Duality between Message Routing and Epidemic Data Replication

Peter Gilbert* Venugopalan Ramasubramanian Patrick Stuedi and Douglas B. Terry
Microsoft Research Silicon Valley, Mountain View, CA 94043

Abstract

Several domains of networking such as Delay Tolerant Networks (DTNs), Mobile Ad hoc Networks (MANETs), and Peer-to-Peer (P2P) networks have the common goal of transporting messages to their intended destinations. These networking domains share many requirements and employ routing protocols that are often composed of similar mechanisms.

In this paper, we explore the synergy between message routing and data replication and propose the use of topology-independent weakly consistent partial replication systems as a messaging substrate for the above networking domains. These replication systems provide the key properties of eventual consistency, disruption tolerance, and at-most-once delivery, which translate into guaranteed delivery, failure resilience, and bandwidth efficiency—three much desired and repeatedly implemented aspects of message routing. We outline an implementation of a messaging system on top of a replication platform with content-based filtering and show that our design is general enough to implement many different routing policies employed in DTNs, MANETs, and P2P networks.

1 Introduction

Message routing in such diverse networking domains as Delay Tolerant Networks (DTN), Mobile Ad Hoc Networks (MANET), and Peer-to-Peer (P2P) networks has common goals. Routing protocols seek to: 1) transport messages from a sender to one or more destinations, 2) deal with temporary or permanent disruptions in communication between two neighboring hosts, and 3) avoid the cost of multiple repeated transmissions of a message to the same host, for instance in the presence of a routing loop. Naturally, routing protocols designed for different environments tend to reuse mechanisms. Controlled, hop-limited flooding, for example, is common in DTN [13], MANET [7], and P2P [14] routing protocols. Similarly, many protocols maintain temporary message histories to avoid forwarding of duplicate messages.

In this paper, we explore the synergy between message routing and weakly consistent, epidemic-style replication [12]. It is easy to think of a message as an item created by the sender that needs to be replicated at its intended destinations, and possibly also at other intermediate routing hosts. The question we consider is whether a weakly consistent replication system can serve as a suitable platform on which to layer messaging systems. A number of replication systems have been developed that provide three useful properties for message routing:

- **Eventual consistency.** This property means that each item will be eventually replicated on all the hosts wishing to hold copies of that item. For a message routing system, this translates into the key property of guaranteed message delivery.
- **Disruption tolerance.** By design, weakly consistent replication systems enable disconnected operation and opportunistic communication leading to high resilience to network failures. This attribute maps well to the fluid network topology of DTN, MANET, and P2P networks, where a host may be connected to a subset of other hosts at one time, a different subset of hosts at a different time, or be completely disconnected from all hosts.
- **At-most-once delivery.** State-of-the-art replication systems maintain metadata called *knowledge* at each host to ensure that an item once replicated on a host will not have to be replicated again, avoiding the cost of duplicate message transmissions. Furthermore, they strive to keep the knowledge compact, contributing to improved efficiency for routing protocols.

In this paper, we make a case for using weakly consistent replication as a substrate to support message routing. While a wide variety of replication protocols have been devised with the three properties outlined above [12], we particularly focus on systems that additionally support topology independence and content-based partial replication. We call these *peer-to-peer filtered replication* (PFR) systems. In such a system, Cimbiosys [10] for example, each host can select the set of items it wants to store through a filter defined on the contents of the items and then change this selection criteria at any time. We show how messages can be implemented as short-lived

*Peter Gilbert is currently affiliated with the Department of Computer Science at Duke University.

items in a PFR system; describe how selection filters can be used to express routing constraints; and discuss the implications of this approach for the replication system and message routing protocols, drawing examples from DTN, MANET, and P2P networks.

2 Overview of PFR systems

A peer-to-peer filtered replication (PFR) system allows a participating host to store only that subset of items of interest to the host. This interest is specified as a content-based *filter*. The filter is a boolean predicate defined on the content of an item. For instance, it could be specified as the WHERE clause of a SQL query, but need not take this form. Applications may change a host's filter at any time so as to discard items currently in the store, to broaden the scope of items they want to store, or both.

As with any weakly consistent replication system [12], PFR systems allow applications on a host to *insert* a new item and *update* or *delete* an existing item, which may have been inserted by a different host. Applications may perform these operations even when the host is disconnected from other replication hosts. The system guarantees **eventual filter consistency** – that is, newly inserted or updated items will eventually be replicated on all hosts whose filters select those items, updates will be applied in the causal order, and deleted items will eventually be expunged from all hosts. In addition, an application may *discard* an item from the local store, which will not expunge the item from the rest of the system.

Replication happens in a PFR system through an epidemic-style, peer-to-peer synchronization protocol, during which hosts exchange updated (and inserted and deleted) items with each other. Synchronization between a pair of hosts may be *opportunistic*, occurring when two hosts come in contact with each other, *periodic*, following a regular pattern, or *update-induced*, whenever one host has a new updated item to send to another host. The replication system need not place any restrictions on the topology formed by synchronizing partners. Even though a connected topology of synchronization partnerships is necessary to achieve eventual consistency, the topology does not have to be fully connected at any particular instant of time. Nor does a host need to synchronize directly with all the other hosts (even eventually). This fluid, **topology-independent** communication model fits well with the intermittently-connected network topology of DTN, MANET, and P2P networks.

PFR systems have been designed to keep the synchronization overhead small. A *source* host sending updated items to the other *target* host during synchronization does not send any item in which the target is not interested, which the target already stores, or for which the target has a more-recent, updated version. Modern weakly consistent replication protocols achieve this property of **at-most-once delivery** by maintaining a representation of the set of versions known to a host in a data structure called *knowledge*. The knowledge representation is concise, with state proportional to the number of hosts in the system rather than the number of items. Hosts first exchange their knowledge and filters before synchronizing updates. Frequently synchronizing partners may avoid exchanging knowledge and filters repeatedly by caching each other's knowledge and filter, exchanging them only when changes occur.

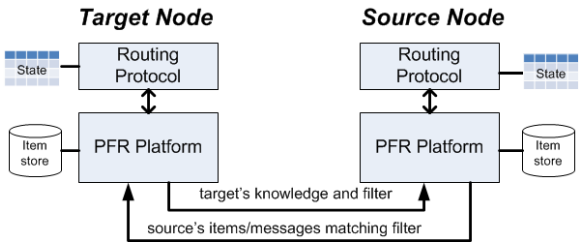

Figure 1: Architecture of a PFR-based message routing system

Cimbiosys [10] is the prototypical example of a weakly consistent replication system that supports content-based partial replication and provides all of the desired characteristics of a PFR platform. Other replication systems such as Bayou [9] and PRACTI [2] provide many but not all of these features.

3 Implementation of a messaging system

In this section, we outline an implementation of a message routing system as an application layered on top of a PFR system. Figure 1 illustrates the architecture described below.

3.1 Message structure

A message, which is stored as a replicated data item, consists of a body or data portion and a header or *metadata* portion. The metadata may include information commonly present in message headers such as a *source address*, *destination addresses*, and *message type*. These are examples of fields whose values do not change for the duration of a message's lifetime in the network. In addition, the metadata may also include additional transient, host-specific information such as a *time-to-live* (TTL) field, which can change as copies of a message are passed among hosts. This distinction between host-specific and host-independent parts of the metadata is important because only changes to the host-independent fields are replicated through the system.

3.2 Synchronization topology

Topology-independent replication systems delegate the responsibility of establishing synchronization partnerships to the application. For a messaging system, syn-

chronizations could happen just between the neighbors in the network topology. Message routing systems employ various mechanisms for neighbor discovery, which they can also rely upon to find potential synchronization partners.

3.3 Filters and routing policies

Hosts define filters to specify the messages they should receive. For instance, a host may define a filter to select any message with the host's address as one of the destination addresses or to select any broadcast message. This flexibility enables the messaging system to support different types of message delivery: 1) unicast messages addressed to a specific destination, 2) multicast messages addressed to multiple destinations, and 3) messages not addressed to a specific destination but to any host meeting certain constraints. Examples of the third category include hosts interested in a particular type of message, hosts with certain attributes (e.g., an ultra-peer in Gnutella [14]), or hosts within a certain number of network hops from the sender (e.g., expanding ring search in AODV [7]).

In addition, many messaging systems enlist hosts to forward messages on behalf of others to enable multi-hop routing. In this case, a host may indicate in its filter the willingness to receive certain messages on behalf of other hosts. Routing protocols typically decide how to forward messages from a source host to a destination host based on different types of information: message-specific metadata such as the source and destination addresses, host-specific message metadata such as the TTL value, and local state maintained by the routing algorithm running on the source and target hosts.

Therefore, in order to accommodate many different routing policies, filters need to be expressive enough to include the following three types of information: 1) metadata attributes specific to a message, 2) metadata attributes that are specific to the host on which the message currently resides, and 3) the state of the source and target hosts participating in the synchronization process.

Note that applications of a PFR system can define their own custom filters, which the synchronization protocol may not understand or parse. Even though the system sends a filter from the target to the source during synchronization, the application, which understands the message and filter formats, performs the final evaluation of whether an item matches the filter. This extensibility enables the message routing system to implement filters in a convenient way. The filter defined by a target host may leave out constraints dependent upon the local routing state at the source host. The source, however, still applies these constraints while evaluating the filter. The advantage of this split implementation of a filter is that changes to the local routing state, which occur frequently in a routing protocol, will not lead to filter changes.

3.4 Message lifetimes

Replication systems retain items permanently until the item is explicitly deleted from the system. While permanent storage might be beneficial for archival purposes, in practice, hosts may run out of storage resources if required to retain messages forever. Therefore, messages in our implementation are as short-lived as possible. A receiver deletes a message immediately after delivering it to the intended application, causing other sites to eventually delete it as well. Alternatively, a message could be deleted by its sender after receiving acknowledgements from the receivers. Intermediate routing hosts can reclaim storage by expunging messages that they have already forwarded. However, they should do so only through the *discard* operation, which removes the message from the local store without deleting the message from the rest of the system.

Note that there is a tradeoff between storage resources and a replication system's ability to propagate messages quickly and reliably. In a resource constrained system, message delivery may be delayed until sufficient resources are available to receive and store the message. A practical approach to deal with storage constraints is to request different delivery guarantees for different types of messages. For instance, small, high-priority, control messages could use the eventual delivery guarantee provided by PFR whereas other (large, low-priority, or data) messages could accept a relaxed guarantee. An example relaxed guarantee is a time-based message lifetime, where the replication system retains the message (and attempts to deliver it) for a fixed time, after which the sending host deletes the message from the system. Choosing an appropriate delivery guarantee is a policy choice.

4 Case studies

A large number of different routing protocols have been proposed for DTN, MANET, and P2P networking domains; the papers [1, 8, 6] respectively provide a good summary of many of them. We examine a few representatives from these domains. First, we discuss issues of reliable routing and duplicate suppression in these protocols, how effective current mechanisms are in addressing these issues, and the potential benefits of implementing them on a PFR system. Later, we show how they can be implemented on top of a PFR platform with example routing filters. Tables 1 and 2 summarize these discussions respectively.

4.1 Delay Tolerant Networks

DTNs are highly-partitioned networks, where contemporaneous end-to-end paths may never exist [4]. DTN routing typically requires taking advantage of intermittent links that form between hosts as they move around. It is not possible to find a single optimal path to route

Routing Protocol		Reliability	Duplicate Suppression
DTN:	Epidemic PROPHET Spray&Wait	multi-path forwarding for all messages (best effort)	table of previously seen message ids
MANET:	AODV	retransmissions of route requests (best effort)	table of previously seen route requests; TTL bounds worst-case duplication
	OLSR	periodic retransmissions of control packets (best effort)	table of per-host counters; TTL bounds worst-case duplication
P2P:	Gnutella	flooding (best effort)	table of forwarded query identifiers; TTL bounds worst-case duplication

Table 1: **Mechanisms used to achieve reliability and duplicate suppression in representative routing protocols.**

a message from its sender to recipients in practical scenarios because in general future contact patterns and network load cannot be predicted with certainty.

As a result, typical routing algorithms, such as Epidemic [13], PROPHET [5], and Spray&Wait [11], which we individually discuss later, flood the network with the message, using some heuristics to control the extent of flooding. These heuristics create opportunities for the message to flow through multiple paths and reach the recipients with high likelihood, but reliable delivery is not guaranteed. Moreover, retransmission protocols are not effective due to large forwarding delays.

Since flooding-based protocols usually result in duplicate message transmissions, these protocols also employ a mechanism to limit duplicates. They store identifiers of recently forwarded messages temporarily in a table and do not forward messages recorded in this table. The effectiveness of this mechanism depends on the size of this table, how long entries are stored, and the lifetime of the messages.

4.2 Mobile Ad Hoc Networks

MANETs provide communication among wireless hosts without relying on any pre-existing infrastructure. MANET routing protocols differ from DTN protocols in the key property that they only use contemporaneous paths and queue up messages only for a brief period of time. They send messages (packets) over a multi-hop route, by either discovering the route on-demand or by maintaining a *routing table* proactively in the background. We examine one candidate from each class: namely, the Ad hoc On-Demand Distance Vector (AODV [7]) protocol and the Optimal Link-State Routing (OLSR [3]) protocol.

As in conventional network routing, these protocols do not try to provide reliable delivery for data packets; reliability is addressed separately at the transport layer if required. However, they do seek to obtain some reliability in the propagation of control messages. For example, when AODV initiates the process to discover a route between two hosts, it repeats the process in order to compensate for the loss of route request or route reply messages. Similarly, when an OLSR host periodically broadcasts its current *link state*, that is its local topology, it adds redundant information onto consecutive broadcasts so that the recipients can recover any lost information from future broadcasts. While these best-effort mechanisms for reliability provide some tolerance to packet loss, better reliability guarantees would be preferable.

Duplicate suppression is important in MANETs for both data and control messages. These protocols also crucially rely on broadcast-based mechanisms for route discovery or link state propagation, and hence have mechanisms to suppress duplicate message propagation. AODV suppresses duplicates by temporarily storing route request messages in a cache and not propagating previously cached messages, and OLSR by storing a sequence number of the last seen link state message from each host and only propagating messages with larger sequence numbers.

Even for data messages, it is essential to suppress duplicate forwarding in the presence of routing loops, which might occur occasionally. Conventional implementations of routing protocols bound the worst-case effects of routing continuously in a loop by setting a hop limit (TTL) for each message. With PFR protocols, however, a routing loop will never be fully traversed as a message previously seen at a host will not be forwarded to the same host a second time.

4.3 Peer-to-Peer Networks

P2P file sharing networks are another class of networks that frequently employ message routing algorithms to find peers with files that match a query (typically composed of some keywords) and route replies back to the querying host. P2P networks differ from DTNs and MANETs in that query messages are not explicitly addressed to specific hosts but contain a *key* or a set of keywords that might find matching *values* in many hosts.

Here, we focus on the popular Gnutella Query Routing Protocol (QRP) [14]. Gnutella employs a controlled (hop-limited) flooding protocol, similar to AODV route

Protocol	Routing Filter	State Transformations
Epidemic	msg.global.to = tgt.addr ∨ msg.local.ttl > 0	tgt: msg.local.ttl –= 1
PROPHET	tgt.table[tgt.addr,msg.global.to] > src.table[src.addr,msg.global.to]	tgt: update predictability table src: update predictability table
Spray&Wait	msg.global.to = tgt.addr ∨ msg.local.copies ≥ 2	tgt: msg.local.copies /= 2 src: msg.local.copies /= 2
AODV	msg.global.type = RREQ ∧ msg.local.ttl > 0 ∨ msg.global.type = RREP ∧ tgt.addr = src.cache[msg.global.rreq_id] ∨ msg.global.type = DATA ∧ tgt.addr = src.table[msg.global.to]	msg.local.ttl –= 1 tgt: update rreq cache update routing table
OLSR	msg.global.type = TC ∧ msg.local.forward ∨ msg.global.type = DATA ∧ tgt.addr = src.table[msg.global.to]	if src.addr ∈ tgt.MPR tgt: msg.local.forward = *true* update routing table
Gnutella	msg.global.type = Query ∧ msg.local.ttl > 0 ∧ (tgt.type = Ultra ∨ (tgt.type = Leaf ∧ tgt.index.match(msg.global.keywords))) ∨ msg.global.type = QueryHit ∧ tgt.addr = src.table[msg.global.q_id]	if msg.glocal.type = Query tgt: msg.local.ttl –= 1 add to routing table

Table 2: **Examples of routing filters and state transformations for representative routing protocols.**

discovery and many DTN protocols, for querying. This protocol typically produces sufficient replies for keys matching popular files; however, for finding rare files, a more reliable method is desirable. For duplicate suppression, Gnutella hosts keep a table of identifiers of query and reply messages to avoid repeat propagation. While this technique prevents a host from receiving duplicate messages from the same host, it does not prevent a host from receiving the same message from different hosts. By building on a PFR system, Gnutella could achieve better reliability so that even rare files could be discovered and duplicate messages eliminated.

4.4 Filters and state transformations

We next show how filters can be composed for expressing routing policies. As previously mentioned, filters specify logical predicates over message metadata and routing state stored at the source and target hosts participating in the synchronization process. Additionally, the source and target hosts may update the message metadata and routing state at the end of synchronization. Table 2 shows the routing filters and state transformations for the six candidate protocols we examined.

Table 2 uses the following simple notation. A message stored by a host is denoted as *msg*. Metadata elements associated with a message are classified as either *global* or *local*, indicating whether the value is host-independent or host-specific, respectively. The source (*src*) and target (*tgt*) hosts have attributes such as addresses and optionally other associated state such as routing tables. We do not attempt to represent each routing policy in exhaustive detail; instead, we use the notation to show that the logic of each protocol can be expressed using the proposed filter model. Some protocols also require the source and target hosts to update local routing state upon synchronization; these updates are performed by the routing application rather than the replication layer. We omit the details of local routing state transformations.

Epidemic routing is one of the earliest proposed approaches to DTN routing. It uses a simple flooding scheme but restricts hop count: messages are replicated during each encounter until a specified TTL expires.

PROPHET uses knowledge of past encounters to estimate the likelihood that a host will be able to deliver a message to another host in the future. A probabilistic metric called *delivery predictability*, $P(a, b) \in [0, 1]$, is maintained for each source a and destination b, indicating the likelihood that a will be able to deliver a message to b. Messages are replicated during each contact, utilizing the delivery predictability metric to limit the extent of replication: a copy of each message is forwarded only if the delivery predictability is greater at the other host.

Spray&Wait, as an alternative to approaches which employ flooding scoped by various thresholds or utilities, injects a fixed number of copies of each message into the network. A host wishing to send a message allocates a fixed number of copies. When a host holding copies of a message encounters another host, it transfers half of its copies of that message as long as it has more than one copy. As a result, each "spraying" corresponds to a binary tree rooted at the message source.

AODV is an on-demand protocol. Whenever a route to a particular host in the MANET is required, AODV issues a route request (RREQ). This message contains the id of the requested host and a fixed TTL value that determines the maximum number of hops the message may travel. During forwarding of the RREQ message each host stores in a cache the id of the host from which it received the message. Once the route request is received by the destination host, a route reply (RREP) message is sent back to the source host on the reverse path, establishing the actual entries in the routing tables of the hosts.

A host issues route requests with increasing TTL values until a route is found or the search is aborted. Data packets are routed according to the routing state established by the returning RREPs.

OLSR is a proactive MANET routing protocol which implements efficient flooding using multipoint relays (MPRs), which retransmit messages on behalf of other hosts. Each host uses its two-hop neighborhood information to select a minimal set of MPRs such that all the hosts in its two-hop neighborhood are reachable. Every host in the network maintains a list of hosts, called the MPR selector set, for which it is an MPR. The host retransmits only those messages received from hosts which have selected it as an MPR. The MPR flooding mechanism is used to spread topology information throughout the MANET. All hosts with a non-empty MPR selector set periodically send out a topology control (TC) message. This message contains the address of the originating host and its MPR selector set. Since every host has an MPR selector set, effectively, the reachability to all the hosts is announced. Thus, each host receives a partial topology graph of the entire network. The shortest path algorithm is then used on this partial graph to calculate optimal routes to all hosts and store it in a table. Data packets are routed by simply looking up the routing table.

Gnutella employs hop-limited, controlled flooding to disseminate *Query* messages containing search keywords and a table-based route back for the *Query Hit* response messages. The most recent Gnutella protocol called Query Routing Protocol (QRP) takes advantage of a tiered structure, where *leaf* peers forward queries to well-provisioned *ultra peers* who in turn flood the query to other *ultra peers*. An ultra peer also maintains an index of keywords for each leaf peer that connects to it and may forward a query to a leaf peer only if the query matches. For routing responses, or Query Hit messages, Gnutella uses a protocol similar to AODV (routing back route replies). Each peer keeps a temporary routing table indicating which peer it received a Query message from and routes the Query Hit message to that peer.

5 Conclusions

The paper shows how principles drawn from weakly consistent replication systems apply to message routing. Both weakly consistent replication systems and message routing systems share the goals of tolerance to disrupted network connections, reliable propagation of messages or updates, and alleviation of duplicate messages (updates). Examination of representatives of these systems indicates that the replication systems employ techniques with well-understood properties for achieving these goals, whereas common DTN, MANET, and P2P routing protocols typically use best-effort heuristics.

In this paper, we explore the notion of implementing message routing protocols as applications layered on peer-to-peer filtered replication (PFR) systems. Such replications systems provide topology independence, which naturally supports dynamic multi-hop topologies, and partial replication, which enables the expression of routing policies as content-based filters. We discuss our implementation architecture in the context of six routing protocols drawn from DTN, MANET, and P2P networks and argue that this approach is feasible and useful.

References

[1] A. Balasubramanian, B. Levine, and A. Venkataramani. DTN routing as a resource allocation problem. In *Proc. of the ACM SIGCOMM Conference*, Kyoto, Japan, Aug. 2007.

[2] N. Belaramani, M. Dahlin, L. Gao, A. Nayate, A. Venkataramani, P. Yalagandula, and J. Zheng. PRACTI replication. In *Proc. of USENIX Symposium on Networked Systems Design and Implementation (NSDI)*, San Jose, CA, May 2006.

[3] T. Clausen and P. Jacquet. Optimized link state routing protocol (OLSR). RFC 3626 (Experimental), Oct. 2003. `http://www.ietf.org/rfc/rfc3626.txt`.

[4] K. Fall. A delay-tolerant network architecture for challenged internets. In *Proc. of the ACM SIGCOMM Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications*, Karlsruhe, Germany, Aug. 2003.

[5] A. Lindgren, A. Doria, and O. Schelen. Probabilistic routing in intermittently connected networks. *SIGMOBILE Mobile Computing and Communication Review*, 7(3), 2004.

[6] E. K. Lua, J. Crowcroft, M. Pias, R. Sharma, and S. Lim. A survey and comparison of peer-to-peer overlay network schemes. *IEEE Communications Surveys & Tutorials*, 7(2), 2005.

[7] C. Perkins, E. Belding-Royer, and S. Das. Ad hoc on-demand distance vector (AODV) routing. RFC 3561, July 2003. `http://www.ietf.org/rfc/rfc3561.txt`.

[8] C. E. Perkins. *Ad Hoc Networking*. Addison-Wesley Professional, 2000.

[9] K. Petersen, M. J. Spreitzer, D. B. Terry, M. M. Theimer, and A. J. Demers. Flexible update propagation for weakly consistent replication. In *Proc. of the ACM Symposium on Operating Systems Principles (SOSP)*, Saint Malo, France, Oct. 1997.

[10] V. Ramasubramanian, T. L. Rodeheffer, D. B. Terry, M. Walraed-Sullivan, T. Wobber, C. C. Marshall, and A. Vahdat. Cimbiosys: A platform for content-based partial replication. In *Proc. of the USENIX Conference on Networked Systems Design and Implementation (NSDI)*, Boston, MA, Apr. 2009.

[11] T. Spyropoulos, K. Psounis, and C. S. Raghavendra. Spray and wait: an efficient routing scheme for intermittently connected mobile networks. In *Proc. of the ACM SIGCOMM Workshop on Delay-Tolerant Networking (WDTN)*, Philadelphia PA, Aug. 2005.

[12] D. B. Terry. *Replicated Data Management for Mobile Computing*. Morgan & Claypool Publishers, 2008.

[13] A. Vahdat and D. Becker. Epidemic routing for partially-connected ad hoc networks. Technical Report CS-2000-06, Duke University, July 2000.

[14] Gnutella protocol specification, July 2009. `http://wiki.limewire.org/index.php?title=GDF`.