Manycast: Exploring the Space Between Anycast and Multicast in Ad Hoc Networks

Casey Carter
Department of Computer Science
University of Illinois, Urbana-Champaign
ccarter@cs.uiuc.edu

Seung Yi
Department of Computer Science
University of Illinois, Urbana-Champaign
seungyi@cs.uiuc.edu

Prashant Ratanchandani
Department of Computer Science
University of Illinois, Urbana-Champaign
ratancha@cs.uiuc.edu

ABSTRACT

The characteristics of ad hoc networks naturally encourage the deployment of distributed services. Although current networks implement group communication methods, they do not support the needs of a mobile client that must locate one or more distributed servers. A client should not need detailed knowledge of network topology in order to choose servers with which it can communicate efficiently.

To this end, manycast is a group communication scheme that enables communication with an arbitrary (client specified) number of group members. Anycast and multicast communication are special cases of manycast in which the target number of group members is one and infinity, respectively. We present manycast and discuss its use as a communication primitive, with specific attention to ad hoc networks. We advocate manycast support at the network layer. A manycast routing protocol enables a client to contact several nearby network nodes that implement a distributed service.

We analyze some approaches to manycast, including some application layer implementations. This evaluation supports our claim that manycast must be implemented in the network layer for effective operation in ad hoc networks. We present several extensions to ad hoc routing protocols that can provide manycast support with minimal implementation effort. Through analysis and extensive simulation, we explore the behavior of these approaches to manycast, finally providing recommendations to implementors.

Categories and Subject Descriptors

C.2.1 [**Computer-Communication Networks**]: Network Architecture and Design—*Wireless Communication*; C.2.2 [**Computer-Communication Networks**]: Network Protocols—*Routing Protocols*; C.2.4 [**Computer-Communication Networks**]: Distributed Systems—*Distributed Applications*

General Terms

Algorithms, Reliability

Keywords

Service location, ad hoc routing, manycast

1. INTRODUCTION

In ad hoc networks, intermittent connectivity—due to mobility, congestion and partitioning—limits the availability of centralized services. Ad hoc networks therefore require distributed services, directly enhancing service availability through server replication. A client may wish to contact multiple servers to mitigate the inherent unreliability of ad hoc networks. Alternatively, the service definition may actually require a client to contact a quorum of servers for the service to maintain coherence. A client that wishes to use such a service must determine the location of the servers, as well as decide which server, or set of servers, it should contact. Although support is available for group communication in ad hoc networks, there is no clear method for a client to locate and communicate with a subset of a distributed group of servers.

IP networking already has support for two different group communication schemes: anycast [21] and multicast [5]. A client communicates with group members by transmitting packets addressed to the group. The two schemes differ in the coverage of a communication. For multicast, all members of the group receive every transmission. Anycast delivers each transmission to a single member the group, the member "closest" to the sender as determined by the network layer. We investigate an additional group communication method, manycast, that allows clients to communicate with an arbitrary number of group members as chosen to optimize successful communication as well as communication overhead.

Essentially, manycast is a group communication paradigm in which one client communicates simultaneously with some threshold number k of servers from the m members of a group. Manycast provides a unique communication challenge. As in anycast, the ideal set of receivers for a particular transmission varies according to its source. In fact, both anycast and multicast are special cases of manycast communication, for $k = 1$ and $k = \infty$ respectively. To support service-oriented communication, manycast should enable efficient short transactional request/response communication between clients and servers, in addition to a one-way dissemination of data as in IP multicast. Due to the dynamic nature of ad hoc networks, we claim the efficient support of this bidirectional one-to-many-to-one communication requires implementation in the network layer.

In this paper, we investigate the problem of providing manycast communication in an ad hoc network. Current research addresses manycast from the application layer in infrastructure networks [14]. Application layer manycast has similar shortcomings to application layer multicast. The lack of complete network knowledge, and the inability to control exactly where a transmission is delivered, make manycast at the application layer inherently less efficient than at the network layer. Rapid topology change in mobile ad hoc networks magnifies these inefficiencies. This is the first work to approach efficient manycast as a routing problem in ad hoc networks. The goal of a manycast routing protocol is to support manycast while optimizing the same objectives used for unicast and/or multicast communication (e.g., minimizing hop count, load balancing or conserving resources).

Our contribution is to classify mechanisms that enable manycast and to motivate the use of manycast to support distributed services in ad hoc networks. We show that manycast routing can provide more efficient service than application layer approaches. We investigate mechanisms enabling manycast communication that could support a complete manycast routing protocol.

In Section 2, we discuss the problem of service location with particular focus on ad hoc networks. We engineer manycast to address these problems in Section 3. Section 4 proposes how manycast can be easily implemented into existing routing protocols. Evaluation of these protocols in Section 5 demonstrates the efficiency of network-layer manycast. We devote Section 6 to future work and conclusions drawn from our study of manycast mechanisms.

2. SERVICE LOCATION IN AD HOC NETWORKS

The problem of service location takes on a slightly different aspect in ad hoc networks. High mobility and frequent partitioning force ad hoc networks to use distributed and replicated services. Applications (i.e., service clients) need a way to discover providers of these distributed services. Manycast can provide for discovery of a good candidate set of service providers and enable efficient communication with those service instances. A short summary of candidate applications/services that could potentially use manycast communication completes the motivation for our work.

2.1 Location and Discovery of Distributed Services

The applications we use every day require infrastructure support to compose disparate components that provide services. Electronic mail is supported by a huge network of post offices and mail forwarding agents. Web browsing is made possible by a delicate lacework of web servers bound together by hyperlinks. These familiar services are also critically reliant on Domain Name Service (DNS) [16] to provide the name-to-address mappings that allow our machines to locate distant services.

Other services, localized in nature, require a degree of robustness against failure. They often perform service discovery through a local broadcast-based protocol, such as the Network Time Protocol [15], or Network Information System (NIS) [25]. These services provide an enabling infrastructure for other services, so it is vital that they be resilient to failures.

The required degree of robustness against failure varies greatly from service to service, and from network to network. For a protocol like NIS that provides access to UNIX userid and password databases, a single broadcast packet is sufficient to provide recovery with a second server for those infrequent occasions when the primary server is down. This application context does not require nuclear-reactor-control quality availability, since failures are rare and the consequences of failure are little more than the occasional user angered over a login delay that takes several seconds longer than expected.

In ad hoc networks, even a low level of availability assurance is hard to achieve. The constant mobility of the network nodes, in conjunction with the rapid variability of wireless communications, means that the network cannot necessarily provide assured communication with any other node at any given time. A service that requires coordination with a single fixed piece of infrastructure suffers from the same problems that a centralized service does in an infrastructure network, but those problems are enhanced by orders of magnitude. Ad hoc networks are even expected to partition frequently, so there may be prolonged periods of service outage in those partitions that do not include the central server.

The only approach that somewhat alleviates the effect of partitioning on availability is replication. Increasing the number of servers in the network increases the probability of availability proportionally. However, increasing the number of servers also creates the problem of maintaining consistency between them. Regardless of any consistency requirement, a client may wish to contact more than one server to increase the likelihood that some server is reachable and can provide service.

2.2 Actual Applications

A number of network applications exist today that use communication that matches the manycast pattern. The manycast paradigm that we present in this paper is specifically designed to support the needs of these applications. These are applications that exist today, and can immediately benefit from manycast.

Even in infrastructure networks, the need to communicate with a fixed number of peers from a larger group occasionally arises. Version 4 of the Network Time Protocol [14], currently under development, includes support for a form of autonomous configuration. In the NTP scheme, a client wishes to locate the three best/nearest servers with which to synchronize its clock. All servers are members of a well-known IP multicast group. Clients locate servers by performing an expanding ring search over the IP multicast tree. Once the server set has been located, clients use a very long-periodic refresh to determine if better/nearer servers have appeared in the network [13]. The NTP mechanism is an application layer manycast that uses an approach termed *Scoped-Multicast* in Section 4.2.8.

The ITTC project (Intrusion Tolerance via Threshold Cryptography [1]) provides tools and an infrastructure for building intrusion tolerant applications. ITTC ensures that the compromise of a few system components does not compromise sensitive security information. Cryptographic keys are distributed across several servers using threshold cryptography. The keys are never reconstructed at a single location. Each server knows only a small part of the secret key, so a client must contact several servers simultaneously.

Research on distributed public key infrastructure was the original motivation for our study of manycast. Cornell On-Line Certification Authority (COCA) [28] introduced a distributed certificate authority for wired networks. Again, authority is distributed across several servers using threshold cryptography, so that a client must contact several simultaneously for certification.

Mobile Certificate Authorities (MOCA) [26] extends the distributed certificate authority approach to wireless ad hoc networks. MOCA enhances an ad hoc routing protocol with new message types to perform certification requests and replies. This approach is essentially a special-purpose manycast, what we call *Unicast* in Section 4.2.5. By providing a manycast communication primitive at the network layer, work such as MOCA could be made independent of the routing protocol.

An often cited application of public key infrastructure in ad hoc networks is secure routing. Given PKI, securing routing protocols a much more tractable problem [27].

2.3 Potential Applications

We envision many other applications in ad hoc networks that could benefit from manycast communication.

Peer-to-peer file sharing systems [4, 6] form overlay networks between end systems by forming a number of point-to-point connections between neighboring nodes. A P2P system for ad hoc networks could efficiently locate a number of nearby peers with which to neighbor in the overlay by using manycast.

A sensor network is a static ad hoc network composed of small devices with sensing capabilities scattered throughout an area of interest. Sensor nodes detect an event in the environment and report this observation either to a data collection point or to observers that move within the network themselves. Manycast could be useful for mobile data collectors that need sensor measurements from the immediate environment. For example, averaging over the 50 closest temperature sensors could provide a reliable idea of the local temperature.

Distributed applications in a ubiquitous computing environment or smart space [24] often need to configure an ensemble of nodes to perform a computation. A bootstrapping application could use manycast to find the k nearest nodes in the smart space capable of providing a particular type of computational service (e.g., the two nearest displays).

In general, many distributed database or cache coherence applications could benefit from manycast communication. Manycast enables each component of such an application to discover a subset of its peers with which it can most efficiently synchronize. By providing this location service in the network layer, manycast isolates the application from knowledge of network characteristics and topology changes. For example, consider the class of geographic routing protocols.

Geographic routing protocols forward packets through the neighbor that is geographically closest to the destination. Greedy Perimeter Stateless Routing (GPSR) [11] achieves routing scalability by assuming the existence of an infrastructure of location database servers from which a client retrieves the location of a destination. Querying the location databases and maintaining database coherence are both interesting target applications for manycast.

3. DEFINING A MANYCAST SERVICE

The need for efficient group-subset communication in ad hoc networks motivates the development of specialized, yet simple, communication support. To this end, we propose a network layer manycast communication primitive. Manycast is a group communication paradigm in which one client communicates simultaneously with k of m equivalent servers in a group. Exactly how the k are chosen from the m is independent of the communication model itself, but may affect the service perceived by applications.

3.1 Manycast

Manycast fills a spectrum of network communication space between anycast and multicast. Like multicast, manycast allows a source to communicate with many destinations simultaneously. As in anycast, the network itself chooses the exact set of destinations from a larger set of candidates. Explicit support for manycast allows an application to target a number of destinations other than 1, as in unicast or anycast, or all, as in multicast.

Manycast provides a bidirectional channel to enable request / response communication between client and servers, not merely one-way dissemination of data. This bidirectional communication follows a one-to-many-to-one model. Like anycast, manycast clients that perform stateful communications with servers will have problems, since the receiver set of any manycast transmission is not fixed. Due to this dynamicity, we expect most uses of manycast to be brief request/response transactions.

The brevity of session lengths for manycast communications encourages the development of reactive, stateless implementations. In the proper circumstances, network-wide broadcast may be a perfectly suitable implementation of manycast delivery. A more proactive solution—finding, establishing, and maintaining an ideal distribution tree in the network—may have overheads that dwarf the cost of a single network flood delivery.

Applications that need a long-term or stateful interaction are best served by using a brief manycast transaction to discover k servers. Those servers can then be contacted using unicast communication, or directed to join a source-specific multicast group. Unicast and multicast communication better support a persistent session.

3.2 Challenges to Manycast

Ad hoc networks create some unique challenges to manycast communication support. Particularly, the combination of ad hoc network characteristics and on-demand operation forces manycast implementation into the network layer.

One can conceivably implement manycast outside the routing protocol. Application layer manycast modules present on each node in the network would then flood a transmission throughout the network to perform manycast delivery. Since an on-demand routing protocol requires a network flood to perform unicast route discovery, this seems like a nice, easy solution without being too expensive. This illusion is quickly shattered when considering the return path. Since the flood request reaches all m servers, each of them unicasts a response to the client. The routing protocol on each server node floods a route request for the client's unicast address to get a return route. In total, the single manycast transaction results in $m + 1$ network-wide floods.

This excess of network-wide floods can have catastrophic ramifications at the MAC layer. The route request floods converge toward the client, resulting in a broadcast implosion—a reverse broadcast storm [18]—possibly causing collisions and loss of many of the responses. Even if the route requests are successful, unicast routing of the responses may trigger another broadcast implosion as a wavefront of broadcast ARP requests surges toward the client. It has been shown that the ARP effect alone can result in nearly 30% loss of responses [3]. For manycast to be efficient and effective, it must be natively supported by the routing protocol.

3.3 Service Quality

Manycast supports a best-effort attempt to reach k servers but provides no guarantee that exactly k are contacted. The quality of a manycast delivery process is evaluated by the proportion of satisfied requests. A request is satisfied when at least k distinct server responses arrive back at the requester. Since 1-to-k communication is k times more likely to fail than is 1-to-1, it is necessary to consider the service quality provided by manycast—the goal is not to provide perfectly reliable transmission, but to perform better than best-effort.

We propose an application interface for manycast that allows a client to specify both parameters k, the number of responses desired, and the desired reliability level. This paper defines reliability as the expectation that any given request will receive at least k responses (i.e., be satisfied). Note that this is unlike the usage of "reliability" in the multicast routing literature to mean guaranteed delivery. An implementation that wishes to improve service reliability may expend resources by contacting $K > k$ servers to increase the likelihood that at least k respond successfully. This approach to reliability variation allows the manycast implementation to adapt to the network, isolating the application itself from communication characteristics. The proper choice of K to provide an application's desired reliability level is implementation-specific, and probably will need to adapt to the network. The simulation study in Section 5 investigates how the choice of K affects the behavior of manycast communication.

Applications with reliability needs that differ from that provided by manycast service can of course extend these reliability semantics at higher layers. Just as TCP [23] provides full reliability for unicast communication, one can imagine transport protocols atop manycast that provide similar functionality. Since not all applications will require this stricter reliability semantic, as is the case for unicast communication, the end-to-end argument justifies the use of simple reliability semantics in the network layer. Exploration of transport layer issues for manycast communication is a topic for future research in this area.

4. MANYCAST ROUTING

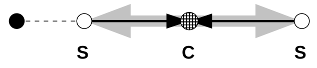
Having motivated the need for manycast support, and the necessity of providing that support in the network layer, the feasibility of manycast routing remains to be shown. Can it be done at all? We present several approaches to manycast support to demonstrate that manycast can be easily implemented into routing protocols, conserving overhead relative to application layer approaches.

4.1 Vocabulary

To provide a clear description of the manycast approaches, we define some necessary terminology and a simple graphical representation of communication patterns. Just as with unicast routing protocols, there are two phases to the routing process. In the *discovery* phase, the source has no knowledge about the network and the targets of a transmission. In the *delivery* phase, the source has previously discovered something about the topology of the network and tries to use that knowledge to perform more efficient delivery than what is possible in the discovery phase. When the network knowledge is no longer useful, e.g., due to a link break, the routing process moves from delivery back into the discovery phase.

We evaluate approaches using the total number of transmissions in a delivery process. Total transmissions is a fair approximation to energy consumption. It is equivalent to hop count in the unicast case. To provide a more detailed explanation of each approach's behavior, we categorize the different types of transmissions necessary to perform a transaction between source and destination nodes. When the source first transmits the request message, we term that transmission a *request*. Any retransmissions of the message by intermediate routers are called *relays*. When a response message is first sent from a server, we term that transmission a *response*. Finally, unicast transmissions that carry the response along the return path between server and client are *forwards*. We also classify transmissions as either link-layer broadcast or link-layer unicast transmissions to characterize the behavior of the schemes.

We illustrate the operation of each approach using the ad hoc network of $n = 28$ nodes in Figure 1. This network supports a manycast application with one client node, the hatched node labeled C, and $m = 11$ servers, the white nodes labeled S. The goal for this application is to reach $k = 3$ servers with each manycast transaction. The dashed lines between nodes indicate connectivity. We indicate request and relay transmissions (those that deliver the request to servers) as thick, lightly shaded directed edges. Transmissions that carry the response message back to the client are depicted as narrower, darker directed edges. A depiction of these edge styles illustrates the representation:



When a transmission is sent in both directions across a link during the course of a scenario, we remove the arrowheads and draw a single undirected edge between the two nodes. We indicate broadcast transmissions with multiple outgoing edges from the same node. The description in the text clearly differentiates whether multiple outgoing edges indicate a single broadcast transmission or several unicast transmissions.

A familiar unicast routing example, illustrating the operation of the Dynamic Source Routing protocol (DSR) [10], clarifies this representation. In the discovery phase of DSR, when a source first needs to send a data packet to a new destination, it floods the network with a route request packet. The destination, upon receiving the flooded request, unicasts a response back to the sender along the reverse of the route that the request followed to reach the destination. This process is depicted in Figure 2. During the delivery phase of operation, the source and destination unicast back and

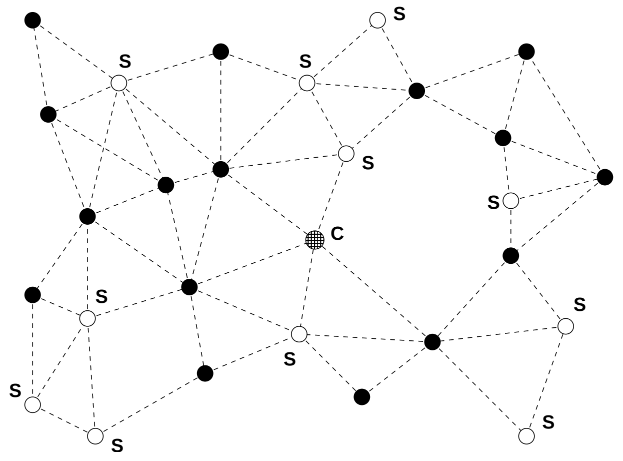


Figure 1: An ad hoc network

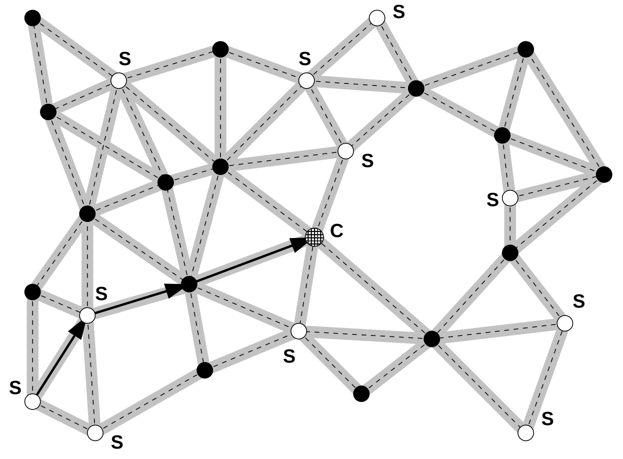


Figure 2: Unicast routing with DSR

forth along the route discovered in the discovery phase. The cost incurred during the discovery phase is

1 request	+	27 relays +
1 response	+	2 forwards
	=	31 total transmissions,

whereas during the delivery phase – actually forwarding data packets – "transaction" cost is only

1 request	+	2 relays +
1 response	+	2 forwards
	=	6 total transmissions.

4.2 Mechanisms

We discuss a set of mechanisms that alter an on-demand ad hoc routing protocol to perform manycast delivery. Beginning with the simplest possible extension, the mechanisms incrementally evolve to more aggressively optimize the number of transmissions. The goal of this process is to show that several implementations of manycast routing exist that provide different trade-offs between performance, reliability and ease of implementation. For each approach, analysis determines the number of transmissions required to complete a single manycast transaction to provide an analytical estimate of raw performance. Later simulation results in Section 5 will make the relative performance of the schemes more concrete, as well as providing an opportunity to study the reliability of each approach.

4.2.1 Application Layer

Analysis of the application layer manycast approach presented in Section 3.2 provides a basis for comparison. In the example network, the request broadcast incurs 1 request and 27 relay transmissions. Each responder performs a route discovery for the origin, as in the DSR example, resulting in a similar broadcast flood with additional unicast transmissions to propagate route replies as well as transaction responses. The total manycast cost is $m + 1$ floods, plus a number of unicasts equal to the sum of twice the hop count from each server to the origin. For the example network,

$m + 1$ requests	+	$27(m + 1)$ relays +
$2m$ responses	+	26 forwards
	=	384 total transmissions.

Clearly, this application layer manycast approach is prohibitively expensive.

4.2.2 Idealized

An ideal manycast delivery has no discovery phase and requires the smallest possible number of transmissions to perform a transaction. For the example network, an ideal manycast delivery is presented in Figure 3. This delivery reaches exactly k servers. All transmissions, with the exception of the original request, are unicast. The cost of this distribution is

1 request	+	1 relay +
3 responses	+	1 forward
	=	6 total transmissions.

We evaluate the raw performance of the following schemes in terms of how closely they approach this ideal.

4.2.3 Flood

The most obvious approach is to integrate support for flood delivery into the routing protocol, as discussed in Section 3.2. Reliable multicast can similarly be supported in highly dynamic networks [7, 19, 20]. For manycast, the routing protocol floods each

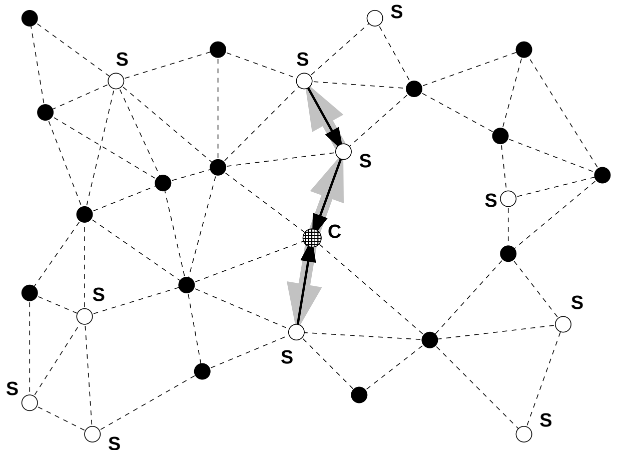


Figure 3: Ideal manycast distribution

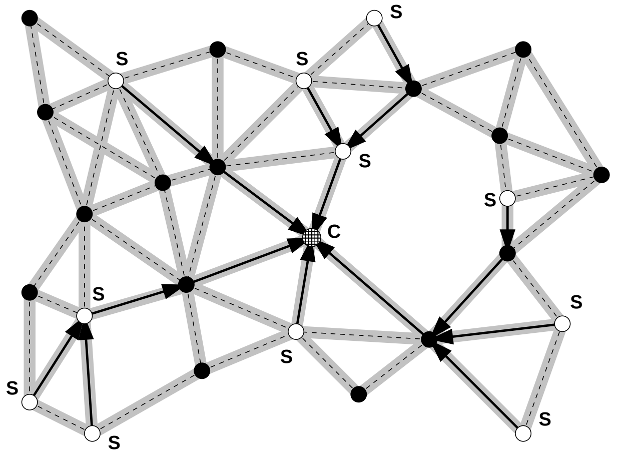


Figure 4: Flood delivery

request, so that intermediate routers set up route state just as they would for a route request. Responses are then unicast along these return routes with no additional discovery process (See Figure 4). The Flood scheme reaches all m servers, but results in m fewer floods of the network than the application layer approach. In the figure,

1 request	+	27 relays +
11 responses	+	13 forwards
	=	52 total transmissions.

Although more efficient than flooding in the application layer, this scheme is still lacking the ability to learn. Sending a network-wide broadcast and receiving m responses back provides many opportunities to disseminate information to and/or collect information from the network nodes for use in later optimizations. The Flood approach, however, is stuck in the discovery phase.

4.2.4 Scoped-Flood

Scoped-Flood is a simple learning extension to the Flood scheme. Scoped-Flood makes one alteration to the operation of the Flood mechanism: it floods the request packet within the smallest TTL-limited scope that covers at least K servers. The discovery phase

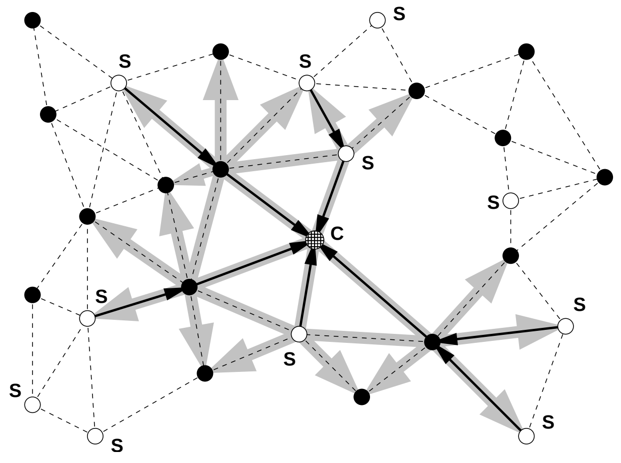


Figure 5: Scoped-Flood delivery with TTL 2

of Scoped-Flood is almost identical to Flood. The routing protocol delivers manycast requests using network-wide broadcast, establishing route state in the network. A slight difference appears on the return path—the unicast replies are treated as route replies by the routing protocol. During future iterations of the delivery phase, the client uses the entries in its route table to find the TTL scope that includes the K closest servers. Figure 5 depicts a delivery phase transaction in the example network using Scoped-Flood. When a request fails to receive K replies, the client returns to the discovery phase. Inspection of the figure yields

$$\begin{array}{lcl} \text{1 request} & + & \text{5 relays +} \\ \text{7 responses} & + & \text{5 forwards} \\ & = & \text{18 total transmissions} \end{array}$$

for the cost of a delivery phase transaction using Scoped-Flood. Trading-off a small amount of state at the source vastly reduces overhead relative to the Flood approach.

When k—and likewise K—is small, this scoping approach tends to localize the request broadcasts in the delivery phase to a small region of the network, ideally containing just K servers. The approach also has a self-healing capability that insulates it against mobility. The servers can move freely within the scope, and when a request is flooded, the unicast routes update automatically.

4.2.5 Unicast

The Scoped-Flood approach maintains unicast routes between the client and at least K servers. This observation leads to another refinement of the approach: during the delivery phase requests are unicast directly to the closest K servers. This approach provides perfect selectivity in the delivery phase; exactly the desired set of servers is contacted. The graphical representation is identical to the ideal delivery depicted in Figure 3, the only difference being that this approach uses several unicasts in place of the broadcast transmissions of the ideal case.

Whether or not Unicast is advantageous over the Scoped-Flood approach depends on network topology, mobility, the value of K and the size of the scope. If the scope is large or K is small, the selectivity of unicast transmissions can require fewer transmissions than complete flooding within the scope. If the scope is small or K is large, flooding may be cheaper. In the figure,

$$\begin{array}{lcl} \text{3 requests} & + & \text{1 relay +} \\ \text{3 responses} & + & \text{1 forward} \\ & = & \text{8 total transmissions.} \end{array}$$

The small value of K in the example makes this approach effective.

4.2.6 Small Group Multicast

Recent work on small group multicast (SGM), also called explicit multicast, enables a source to multicast a packet to an explicitly specified set of destinations [2, 9]. SGM packets contain a list of destination addresses and are propagated through the network along unicast routes. We devise two variants of SGM-based manycast, termed SGM and SGMB (for “SGM-Broadcast”). SGM routers partition the set of destinations by next hop and link-layer unicast a replica of the packet payload through each next hop. To reduce the number of request transmissions to at most one per node, SGMB takes advantage of the broadcast nature of the wireless medium.

The SGM approach to manycast eliminates some of the overhead of the Unicast approach delivery phase by combining several unicasts into a single SGM request packet. This approach is as selective as Unicast and also has the same visual representation (Figure 3). The difference is that each node performs at most one transmission per next hop to propagate requests toward servers. For the example, this translates to

$$\begin{array}{lcl} \text{2 requests} & + & \text{1 relay +} \\ \text{3 responses} & + & \text{1 forward} \\ & = & \text{7 total transmissions,} \end{array}$$

a very slight improvement over Unicast.

For the more aggressive variant SGMB, a packet header contains a list of (next hop, destination) address pairs and is sent via link-layer broadcast. Receivers of the broadcast transmission that are listed as next hops then forward the packet toward the specified destinations. In the example, SGMB reduces overhead to

$$\begin{array}{lcl} \text{1 request} & + & \text{1 relay +} \\ \text{3 responses} & + & \text{1 forward} \\ & = & \text{6 total transmissions,} \end{array}$$

although the optimization can be expected to have much greater impact for larger values of K.

Note that the delivery phase of SGMB achieves the same overhead as the ideal approach, at least in the example. In theory, the optimal distribution tree is not necessarily a tree that covers the closest K servers, but the approach provides a good approximation in practice.

4.2.7 Multicast

An approach that uses traditional multicast routing to perform manycast delivery provides an interesting reference. A multicast tree is more selective than flooding and should therefore put significantly less strain on the network, as a comparison of this approach in Figure 6 with the pure-flooding approach of Figure 4 makes clear. The unicast and multicast routing protocols must be coupled so that multicast request delivery establishes unicast routes in the network for routing reply messages. The cost evaluation of this approach for the example is

$$\begin{array}{lcl} \text{1 request} & + & \text{7 relays +} \\ \text{11 responses} & + & \text{13 forwards} \\ & = & \text{32 total transmissions.} \end{array}$$

This discussion neglects the overhead of maintaining the multicast tree, focusing only on manycast overhead. Simulation results in Section 5 do include maintenance overheads, which must be considered to determine the efficiency of a complete solution.

Although this approach has lower overhead than Flood, it can still be easily improved. Just as for the Flood approach, Multicast is permanently trapped in the discovery phase.

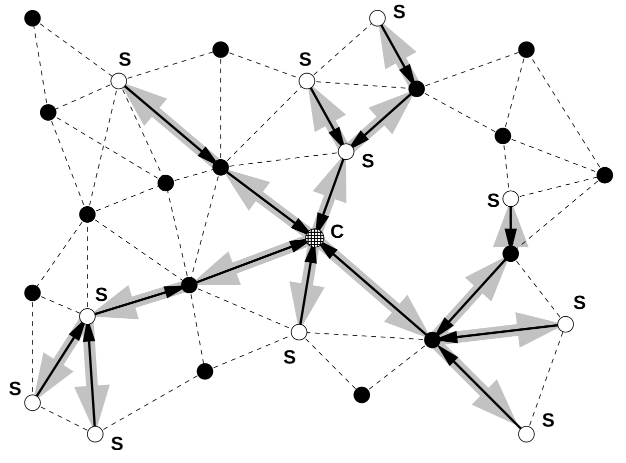


Figure 6: Multicast delivery

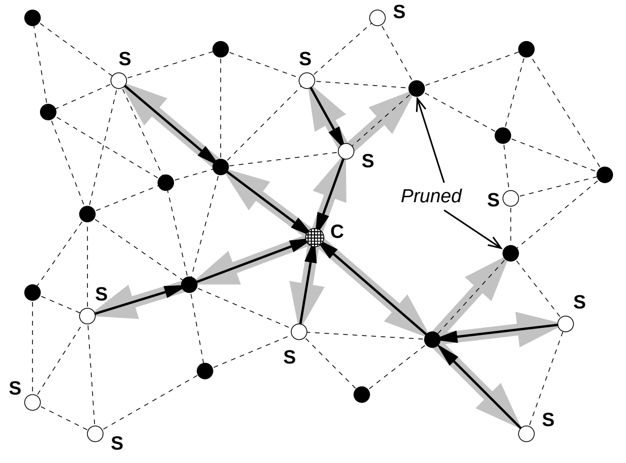


Figure 7: Scoped-Multicast delivery

4.2.8 *Scoped-Multicast*

The application of TTL scoping to transmissions sent via the multicast tree results in Scoped-Multicast, completing the set of mechanisms. Like Scoped-Flood, Scoped-Multicast results in some inefficiency, since the multicast distribution tree may unnecessarily propagate requests down branches of the tree that do not reach a server within the scope.

Figure 7 depicts a hypothetical Scoped-Multicast delivery. To reach K servers in the example, the client must transmit a multicast message with a TTL of two, incurring total costs of

1 request	+	4 relays +
7 responses	+	5 forwards
	=	17 total transmissions.

Ideally, the routers that drop the packet when its TTL reaches zero prune themselves from the tree, as annotated in the Figure. These pruned branches need to automatically re-join the tree when the manycast client returns to the non-TTL-scoped discovery phase. To the best of our knowledge, no current manycast routing protocol has this particular behavior.

Just as Multicast distribution is more selective than Flood, so is Scoped-Multicast more selective than Scoped-Flood. It remains to be seen if the overhead of tree maintenance will destroy this advantage in practice. The overhead of reaching too many servers, however, is not addressed by this approach; Scoped-Multicast lacks the selectivity of the SGM-based approaches.

5. EVALUATION

To evaluate the different mechanisms, we implemented them in the NS2 [17] simulator. The results of simulation provide many insights into the trade-off between implementation complexity, state kept, performance and reliability of each approach.

5.1 Implementation

We extend the Ad hoc On-demand Distance Vector routing protocol (AODV) [22] implementation in NS2 to perform the non-multicast-tree based manycast approaches: Flood, Scoped-Flood, Unicast, SGM and SGMB. Our extension enables AODV to recognize a set of network addresses that refer to manycast groups. Packets sent to manycast group addresses are handled by the manycast delivery mechanism. AODV creates route state for manycast request/reply packets just as it does for its own route request/reply packets. Route table entries distinguish manycast servers from non-server nodes; the manycast process is in delivery phase whenever K routes to servers are available. For the TTL-scoped delivery mechanisms, we add a method to AODV's route table for determining the smallest TTL that will reach K servers. The addition of an SGM forwarding mechanism enables AODV to support SGM and SGMB delivery. For our testing, AODV was configured to use 802.11 delivery failure notifications as indications of link breakage.

The routing protocol in the multicast-based schemes is Adaptive Demand-driven Multicast Routing (ADMR) [8]. ADMR operates almost entirely on demand, building source-specific multicast distribution trees by flooding a source's first multicast data packet to a group address. Forwarding nodes overhear their children in the tree rebroadcasting data packets, which serves as a passive acknowledgment. After forwarding some number of packets without receiving any passive acknowledgments, nodes automatically prune themselves from the tree. For each distribution tree, the source node maintains an estimate of the inter-packet transmission time for that tree. If no packet arrives after a significant time, the estimate is increased and a keepalive packet is sent down the tree with the new estimate. Non-source nodes on the tree use the inter-packet estimate to decide if a lack of data packets indicates that they have become disconnected from the tree, so that they can perform tree repair.

The implementations of AODV and ADMR within NS2 are coupled tightly so that ADMR multicast dissemination creates unicast route state within AODV. This coupling ensures that AODV unicast routes are in place to forward manycast replies at the time the ADMR multicast requests are delivered, i.e., no additional route discovery process is necessary to deliver replies.

5.2 Simulation Study

All simulations were run in an ad hoc network consisting of $n = 150$ nodes spread uniformly through a 1000x1000 meter square area. Nodes are equipped with an IEEE 802.11 radio network interface, operating at 2Mbps data rate with a 250m transmission range. Nodes move according to the Random Waypoint mobility model, with speed uniformly distributed between 0 and 10m/s and pause-time of 10 seconds. Simulations run for 600 seconds, during which 100 non-server nodes generate 10 64-byte requests each. There are $m = 30$ manycast servers in the network, unless otherwise specified.

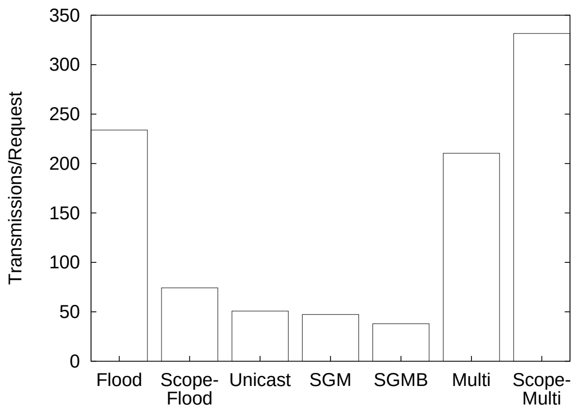


Figure 8: Packet transmissions per request, simple scenario

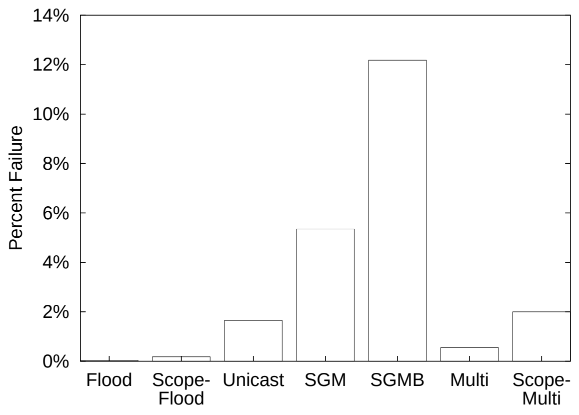


Figure 9: Percent of failed requests, simple scenario

The number of packet transmissions at the router level indicates performance in each set of simulation runs. Each hop-by-hop transmission of the same packet is counted independently. This counting method agrees with that used in the analysis in Section 4.2. The number of satisfied requests serves as reliability metric.

5.3 Mechanisms

The intent of the first set of simulations is to differentiate the behavior of the various mechanisms, to provide a first notion of performance and reliability. These runs use the most deterministic traffic scenario possible: in turn, each client generates its 10 requests with a constant inter-request time of 0.6 seconds – the first client sends its 10 requests starting at time 0, the second client begins at time 6, and so on. This traffic scenario maximizes the locality of the requests in the network, allowing the aggressive optimizations to shine. Although this traffic is not necessarily characteristic of any particular service in ad hoc networks, it does provide a very simple case to examine the mechanisms themselves.

These simulations set the parameters $k = 9$ and $K = 10$, i.e., each request attempts to contact 10 servers and is considered satisfied upon receiving the 9th response. Each data point is averaged over 10 separate mobility scenarios with the same parameters.

Figure 8 reports the total number of packet transmissions generated by each approach. Note that for the non-ADMR runs, the more aggressive approaches do reduce the total number of transmissions as expected, although perhaps not as much as expected. In these simulations, there are almost always K servers within two hops of a client node, so Scoped-Flood is very effective at reducing overhead. The more aggressive optimizations show little improvement over Scoped-Flood since there are not many redundant transmissions for them to eliminate within a two-hop scope.

One anomaly here is the lackluster performance of the ADMR-based approaches: Multicast, which barely outperforms Flood, and Scoped-Multicast, which is much worse. Some insight into the operation of ADMR explains this behavior. At simulation startup time, all servers join the manycast group. Each node's ADMR module floods a "Multicast Solicitation" packet through the network. The ADMR trees are slow to die, keepalive packets flow down each tree for some time after the source stops sending requests. For example, in one scenario, ADMR generated 72,000 control packet transmissions. The problem here is not a deficiency in ADMR itself so much as a mismatch between the objectives of ADMR, which is optimized for a very small number of long-lived multicast flows, and manycast.

The mismatch between manycast and Scoped ADMR is even more pronounced. ADMR reacts catastrophically to the TTL limit. Nodes on the tree but outside the delivery scope treat the lack of data and keepalive packets as an indication of tree disconnection. Each subtree outside the delivery scope begins to flood the network with tree repair packets. In one Scoped-Multicast simulation run, ADMR generates 244,000 transmissions of non-data packets. The effect of scoping on ADMR is to increase the overhead nearly fourfold.

The lessons to learn from the failure to integrate ADMR with manycast are 1) distribution-tree oriented routing protocols need to be specially designed to support manycast, and 2) it is even more important than we first realized to support manycast in the network layer. Applications using Scoped-Multicast in the application layer would devastate a network that uses ADMR.

The performance numbers alone do not tell the whole story. Figure 9 reports reliability, in terms of the percentage of unsatisfied requests. The reliability of each approach is inversely related to its performance, since eliminating redundant transmissions lowers reliability in the presence of losses. Flood is very reliable, with a failure rate of about 0.03%, and Scoped-Flood is very successful at eliminating transmissions without inducing additional error, with a failure rate of about 0.2%.

The Unicast approach, at 1.7%, shows mobility-induced losses, which are significantly amplified by both SGM and SGMB. The higher failure rate of SGM vs. Unicast is caused by the fate sharing of all servers reached through a particular next hop. SGMB's increase over the failure rate of SGM is also significant, and caused by 802.11's lack of RTS/CTS/ACK protection for broadcast transmissions. The absence of RTS/CTS causes an increase in collision-induced losses, while the absence of ACKs means that AODV cannot detect link breaks. Intuitively, use of SGMB should lower the congestion of the medium relative to SGM by a factor of the average number of neighbors. However, the effect of this benefit is lost, since SGMB is much more susceptible to congestive losses. The net effect is an overall decrease in efficiency for SGMB.

For the remainder of the section, we introduce a new metric that couples performance and reliability: normalized transmission cost, the total number of packet transmissions per satisfied manycast request. Figure 10 displays this normalized metric for the approaches in this set of simulations. Relative to the raw performance numbers in Figure 8, the Scoped-Flood, Unicast, SGM, and SGMB approaches are much closer in terms of normalized cost – increased reliability can be expensive. Again, poor performance of ADMR

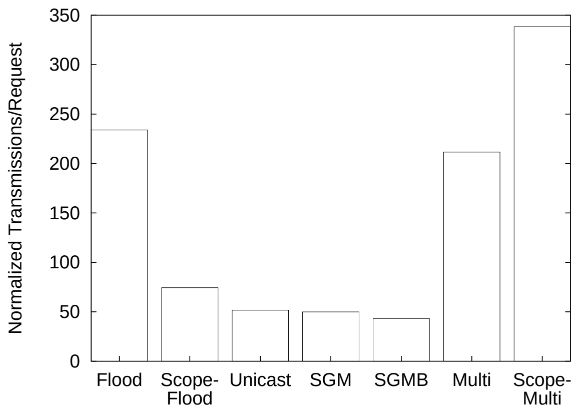


Figure 10: Transmissions per satisfied request, simple scenario

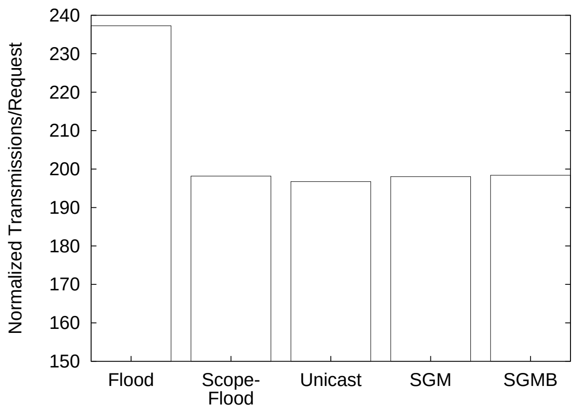


Figure 11: Transmissions per satisfied request, complex scenario

is due to the mismatch between the design goals of ADMR and manycast. For this reason, the remainder of the paper does not report ADMR-based simulation results.

5.4 Complex Traffic

Having obtained a basic understanding of how the mechanisms relate to each other through the simple traffic scenario, the next logical step is to study a more complex situation. In the "complex" traffic scenario, each of 100 mobile nodes still makes 10 requests, but at times distributed uniformly at random throughout the 600 second simulation run. This increases the mean inter-request time to the extent that temporal locality is destroyed and route caching becomes less effective. This more complex scenario stresses the manycast mechanisms. Each data point is again averaged over 10 runs, with the same parameter values $k = 9$ and $K = 10$.

Normalized request cost is reported in Figure 11 (note the scale is different than in Figure 10). All of the results have been compressed into a small range of the chart, between 196.8 - 198.4 packets transmitted per satisfied request. This scenario's decrease in locality causes all approaches to spend a significantly greater portion of the simulation in the discovery phase. The increased use of Flood discovery raises all approaches into the 200 pkts/request range. Clearly, performance alone is not enough to discriminate between the approaches in this scenario. The raw performance and reliability numbers in Figure 12 and Figure 13 make it clear that

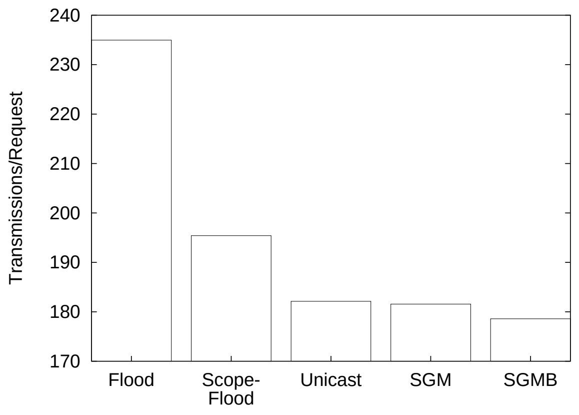


Figure 12: Transmissions per request, complex scenario

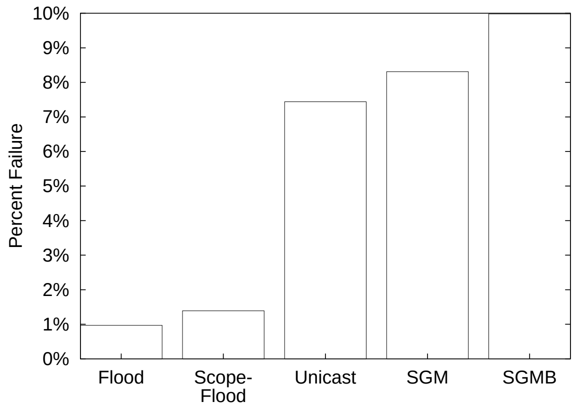


Figure 13: Percent of failed requests, complex scenario

Scoped-Flood has better reliability, while SGMB is the top performer. To complete the picture of manycast mechanisms, it is necessary to investigate both reliability and performance when the parameters m and k are varied.

5.5 Reliability

Since the proposed manycast application interface allows applications to specify a desired reliability level, it is important to investigate the reliability each mechanism provides. In the complex and simple traffic experiments, failure rates vary between 0.03% and 12.2%. The behavior of the manycast mechanisms when failure rate is held approximately constant is important, since failure rates much greater or much less than required by the application can both result in bad performance. This set of experiments uses the complex traffic scenario, with k still 9, and the smallest value of $K \geq k$ for each mechanism that results in an error rate of $5\% \pm 1\%$. The exact values of K, determined empirically, are

Scoped-Flood	Unicast	SGM	SGMB
9	11	11	13

resulting in the well-controlled failure rates in Figure 14. Note that the Flood approach is independent of the choice of K, so its results here are identical to the previous experiment. Scoped-Flood, on the other hand, maintains good reliability with $K = k$ in this experiment. If this reliability holds over a wide range of values for k, as Section 5.7 investigates, then the Scoped-Flood has an advantage

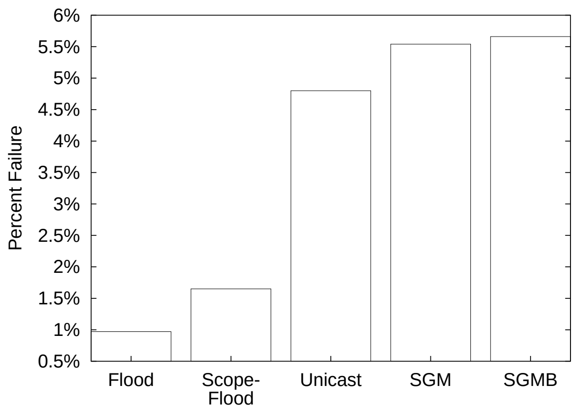


Figure 14: Percent of failed requests, fixed 5% reliability

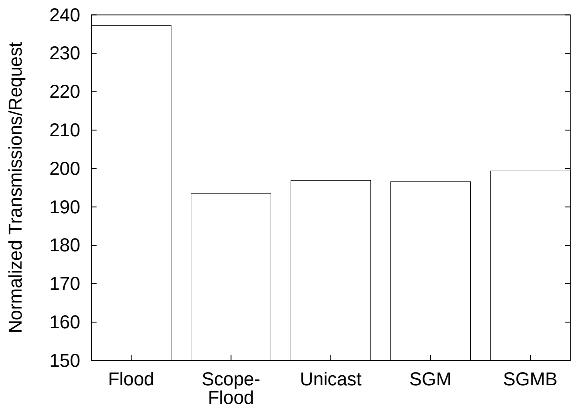


Figure 15: Transmissions per satisfied request, fixed 5% reliability

in terms of implementation complexity: it avoids the complexity of adapting K.

The performance achieved at this reliability level is presented in Figure 15. Scoped-Flood is the most efficient. Notably, there is still little spread in the performance of the different approaches. For comparison, a 10% reliability target gives the same results as in the complex traffic experiment, with the exception of Scoped-Flood, which uses the same parameters as for 5%. At 10% reliability Scoped-Flood is still the most efficient approach, but only by a very narrow margin. The more aggressive approaches pay in reliability for their advantages in performance, resulting in almost equivalent normalized performance numbers.

The end result of the investigation into reliability is to discover that, at reasonable reliability levels of 5-10%, Scoped-Flood is preferred. Just as in earlier experiments, the spread of performance levels is narrow, implying that further examinations are necessary to truly determine the best approach. For the remainder of the simulations, K is set independently for each approach to maintain the 10% reliability level, with a 1% tolerance.

5.6 Varying Server Density

The effect of varying the parameter m, the number of manycast servers in the network, is a key behavioral characteristic. This study determines how the mechanisms react to changes in m, as well as providing insight for designers of manycast applications. How many servers should there be in a network? Intuitively, it seems that more servers will enhance availability and performance by encouraging spatial locality in communication.

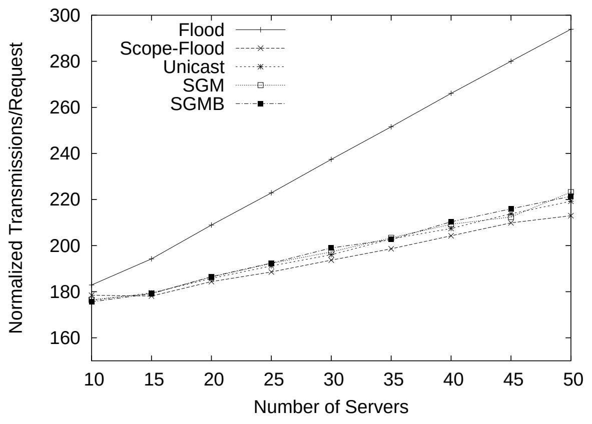


Figure 16: Transmissions per satisfied request, varying server density

These experiments use the complex traffic scenario and vary the number of manycast servers m between 10 and 50 by increments of 5, with k again fixed at 9. K is independently chosen for each combination of approach and m, to bound the error rate at the 10% reliability level. The resulting empirically determined values of K are

Scoped-Flood	Unicast	SGM	SGMB
9	10	10	10, $m < 40$
			11, $m \geq 40$

For higher values of m, SGMB needs a greater K to protect its requests against congestion.

The normalized performance in Figure 16 reports the first set of simulation results. There is an interesting lesson here for the ad hoc network architect: increasing the number of servers in the network has a direct negative consequence on performance. Clearly, Flood request overhead is unrelated to m, but reply overhead scales directly with m. Since the other approaches are critically dependent on flooding for their discovery phases, their behavior also shows this linear increase with m, albeit with less impact than on the Flood approach itself.

Again, there is little spread in the normalized costs, with Scoped-Flood using its self-healing ability to good advantage. In fact, this characteristic of Scoped-Flood dominates the behavior, as can be seen from the raw transmission counts in Figure 17 (note that the results for Flood have been removed for clarity). Regardless of the value of m, SGMB always uses the fewest transmissions and Scoped-Flood the most. Scoped-Flood's self-healing ability provides greater reliability so that it spends more time in the delivery phase, achieving the best normalized performance.

5.7 Varying k

The last critical facet of manycast behavior is reaction to varying the parameter k. This experiment also uses the complex traffic scenario and varies k from 1 to 24. Tests with $k = 30$ produced catastrophic failure rates of over 60% for all approaches – manycast is not appropriate for reliable transactional communication with a large group. K is independently chosen for each combination of approach and k to bound the error rate at the 10% reliability level. The ideal values of K, again determined empirically, vary substantially with k, as depicted in Figure 18. The failure rates that result

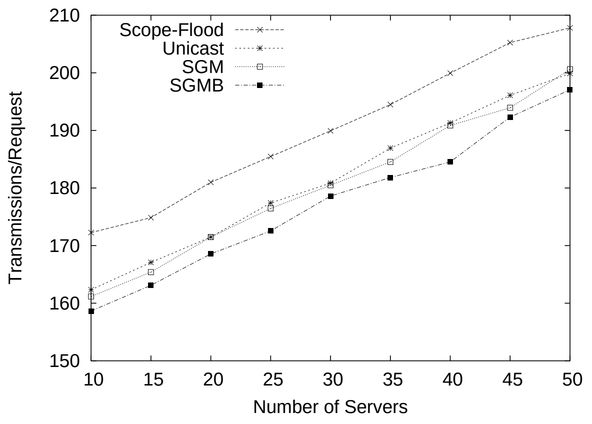


Figure 17: Transmissions per request, varying server density

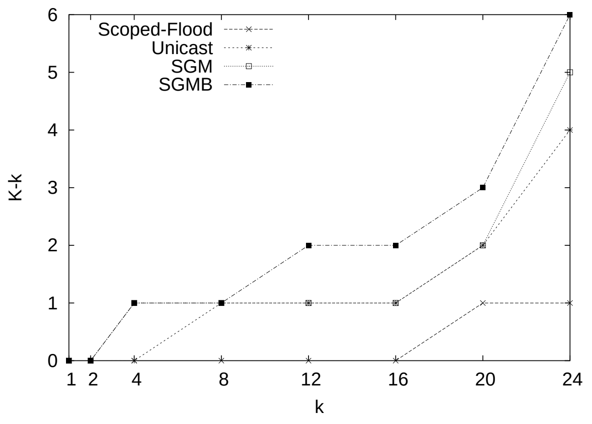


Figure 18: The cost of reliability, $K - k$ vs. k

from these choices of K are in Figure 19. Note that this is the first experiment that required a $K > k$ for Scoped-Flood. From the wide variation in values for K, we infer that an actual manycast implementation needs to dynamically adapt K to observed network characteristics.

The raw and normalized transmission statistics for this set of simulations in Figure 20 and Figure 21 tell an interesting tale. Since the Flood approach operates independent of k, its raw performance is flat, although normalized performance does suffer from lost responses for high values of k. Other approaches asymptotically approach the Flood mechanism as k increases. The reason for this asymptotic behavior can be seen with the assistance of Figure 22, which shows the number of delivery phase (vs. discovery phase) transactions for each approach and value of k. Clearly, as k increases, the manycast approaches spend more time in the discovery phase maintaining reliability, and not enough time in the delivery phase reducing overhead. Again, the same trend holds that was seen in earlier experiments: SGMB uses the fewest transmissions, and Scoped-Flood the most, but Scoped-Flood's higher reliability makes it the most efficient approach. Overall, Scoped-Flood's behavior is clearly the most desirable across the spectrum of simulated experiments. It is convenient to future implementors that one of the simplest mechanisms to implement is the most effective across a wide range of scenarios.

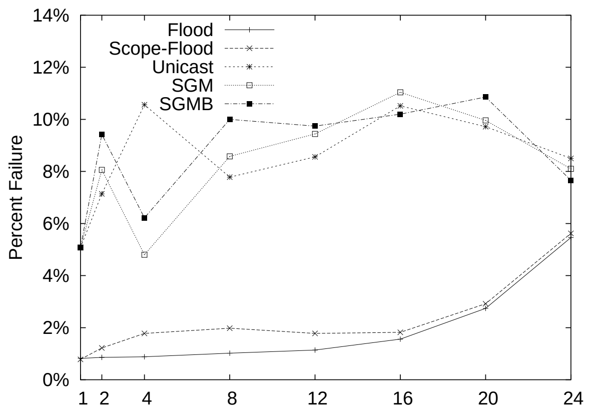


Figure 19: Percent of failed requests, varying k

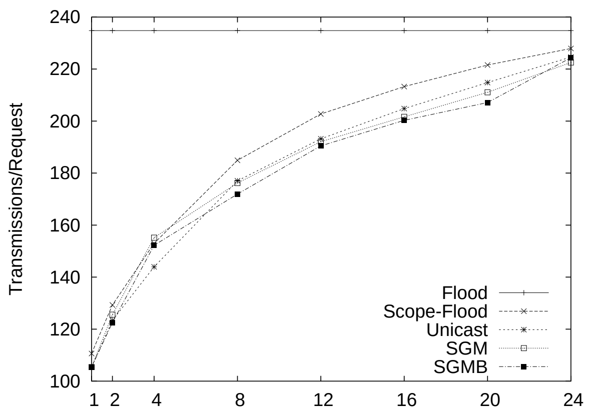


Figure 20: Transmissions per request, varying k

6. CONCLUSIONS AND FUTURE DIRECTIONS

Although none of the proposed simple approaches comes close to achieving the ideal performance of Section 4.2.2, they all significantly outperform the application layer approach of Section 4.2.1. To illustrate, with $K = 4$ and $m = 30$, Flood generates 235 transmissions per request in the simulated network—the application layer approach in Section 4.2.1 would need more than $n(m + 1) = 4650$. Even Flood, the least aggressive but simplest approach to implement, improves overhead by an order of magnitude over the application layer approach. Clearly, manycast is viable and necessarily must be implemented at the network layer.

The simulation study herein shows that the behavior of the proposed manycast mechanisms is dominated by the cost of the Flood discovery phase. Also, the best behaved approach is Scoped-Flood, which keeps some of the Flood approach's insulation against mobility, but at a huge savings in overhead. These two facts imply that a combination of one of the more aggressive approaches (SGM or Unicast) with Scoped-Flood for delivery tree discovery and repair would be most effective. Using the well-known expanding ring search technique from ad hoc routing would enable just such an integration.

Future work is needed to study how a manycast implementation should choose the parameter K to ensure reliability without sacrificing performance. Our simulation study shows that Scoped-Flood

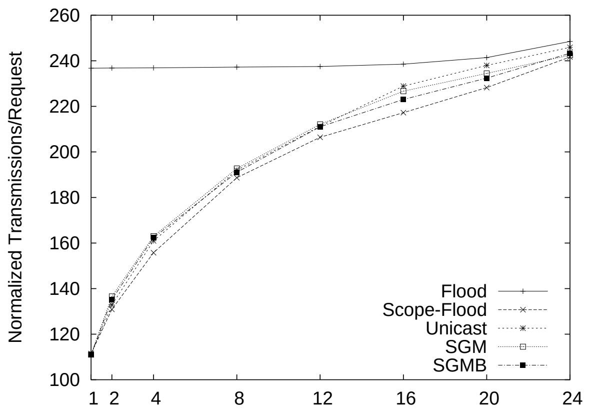


Figure 21: Transmissions per satisfied request, varying k

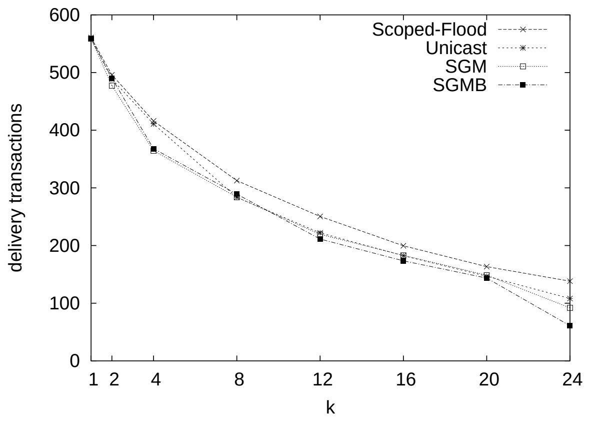


Figure 22: Number of delivery phase transactions, varying k

is effective with K very close to k over a wide range, while other approaches are more sensitive. Also, future research may determine how manycast routing can take advantage of cache sharing. Perhaps the expanding-ring-search approach proposed in this section could benefit from manycast server routes cached at intermediate nodes.

Kozat and Tassiulas [12] propose a virtual backbone approach for service discovery as well as routing in ad hoc networks. Comparison of their proactive approach with the reactive approaches presented herein is an interesting avenue of future investigation.

The interaction between manycast and higher communication layers, especially the transport layer, is an area that we have not yet begun to explore. Application use of manycast for data transport must address congestion control. Just like multicast, a manycast data flow branches out across multiple paths in the network with the potential to cause significantly more congestion than a single unicast flow. It is unclear how retransmission of failed requests will interact with the different mechanisms we present for manycast transport.

7. ACKNOWLEDGEMENTS

We would like to thank:

- The anonymous reviewers of this paper
- Alex Snoeren, for his input as sheperd
- Our colleagues in the Möbius group, for slogging through earlier revisions of this paper
- The authors of NS-2, the simulator that plagues our waking hours
- The first author's wife, for reasons too numerous to list

This work was made possible by funding from National Science Foundation grants ANI-0081308 and CCR-0086094.

8. ADDITIONAL AUTHORS

Robin Kravets, Department of Computer Science, University of Illinois at Urbana-Champaign, email: rhk@cs.uiuc.edu.

9. REFERENCES

[1] T. W. amd M. Malkin and D. Boneh. Building intrusion tolerant applications. In *Proceedings of the 8th USENIX Security Symposium*, 1999.

[2] R. Boivie, N. Feldman, Y. Imai, W. Livens, D. Ooms, and O. Paridaens. Explicit multicast (Xcast) basic specification. Internet Draft (Work in Progress) draft-ooms-xcast-basic-spec-04.txt, Internet Engineering Task Force, January 2003.

[3] C. Carter, S. Yi, and R. Kravets. ARP considered harmful: Manycast transactions in ad hoc networks. In *Proceedings of the IEEE Wireless Communications and Networking Conference (WCNC)*, 2003.

[4] I. Clarke, O. Sandberg, B. Wiley, and T. W. Hong. Freenet: A distributed anonymous information storage and retrieval system. *Lecture Notes in Computer Science*, 2009:46+, 2001.

[5] S. Deering. Host extensions for IP multicasting. Request for Comments (Standard) RFC 1112, Internet Engineering Task Force, August 1989.

[6] Gnutella peer-to-peer file sharing system. http://www.gnutella.com.

[7] C. Ho, K. Obraczka, G. Tsudik, and K. Viswanath. Flooding for reliable multicast in multi-hop ad hoc networks. In *Proceedings of the 3rd International Workshop on Discrete Algorithms and Methods for Mobile Computing and Communications*, pages 64–71, August 1999.

[8] J. G. Jetcheva and D. B. Johnson. Adaptive demand-driven multicast routing in multi-hop wireless ad hoc networks. In *Proc. of the ACM Symposium on Mobile Ad Hoc Networking & Computing (MobiHOC '01)*, Long Beach, CA, October 2001.

[9] L. Ji and M. S. Corson. Differential destination multicast–a MANET multicast routing protocol for small groups. In *Proceedings of IEEE INFOCOM*, April 2001.

[10] D. B. Johnson, D. A. Maltz, and Y.-C. Hu. The dynamic source routing protocol for mobile ad hoc networks (DSR). Internet Draft (Work in Progress) draft-ietf-manet-dsr-08.txt, Internet Engineering Task Force, February 2003.

[11] B. Karp and H. T. Kung. GPSR: Greedy perimeter stateless routing for wireless networks. In *Proceedings of the Sixth Annual ACM/IEEE International Conference on Mobile Computing and Networking (MobiCom 2000)*, pages 243–254, August 2000.

[12] U. C. Kozat and L. Tassiulas. Network layer support for service discovery in mobile ad hoc networks. In *Proceedings of IEEE INFOCOM 03*, 2003.

[13] D. L. Mills. Automatic NTP configuration options. `http://www.eecis.udel.edu/˜mills/ntp/-html/manyopt.html`.

[14] D. L. Mills. The network time protocol (NTP) distribution. `http://www.eecis.udel.edu/˜mills/ntp/-html/`.

[15] D. L. Mills. Network time protocol (version 3). Request for Comments (Draft Standard) RFC 1305, Internet Engineering Task Force, March 1992.

[16] P. Mockapetris. Domain names - implementation and specification. Request for Comments (Standard) RFC 1035, Internet Engineering Task Force, November 1987.

[17] Network simulator 2. `http://www.isi.edu/nsnam/ns/`.

[18] S.-Y. Ni, Y.-C. Tseng, Y.-S. Chen, and J.-P. Sheu. The broadcast problem in a mobile ad hoc network. In *MOBICOM '99*, pages 151–162, 1999.

[19] K. Obraczka, G. Tsudik, and K. Viswanath. Pushing the limits of multicast in ad hoc networks. Technical Report 00-735, USC Computer Science Department, June 2000.

[20] K. Obraczka, G. Tsudik, and K. Viswanath. Pushing the limits of multicast in ad hoc networks. In *Proceedings of the 21st International Conference on Distributed Computing Systems (ICDCS)*, pages 719–722, April 2001.

[21] C. Partidge, T. Mendez, and W. Milliken. Host anycasting service. Request for Comments (Informational) RFC 1546, Internet Engineering Task Force, November 1993.

[22] C. E. Perkins, E. M. Belding-Royer, and S. R. Das. Ad hoc on-demand distance vector (AODV) routing. Internet Draft (Work in Progress) `draft-ietf-manet-aodv-13.txt`, Internet Engineering Task Force, February 2003.

[23] J. Postel. Transmission control protocol. Request for Comments (Standard) RFC 793, Internet Engineering Task Force, September 1981.

[24] M. Román, C. K. Hess, R. Cerqueira, A. Ranganathan, R. H. Campbell, and K. Nahrstedt. Gaia: A middleware infrastructure to enable active spaces. *IEEE Pervasive Computing*, pages 74–83, Oct-Dec 2002.

[25] Sun Microsystems. *System and Network Administration*. March 1990.

[26] S. Yi and R. Kravets. MOCA: Mobile certificate authority for wireless ad hoc networks. In *2nd Annual PKI Research Workshop (PKI03)*, April 2003.

[27] L. Zhou and Z. J. Haas. Securing ad hoc networks. *IEEE Network Magazine*, November 1999.

[28] L. Zhou, F. B. Schneider, and R. van Renesse. COCA: A secure distributed on-line certification authority. *ACM Transactions on Computer Systems*, 20(4):329–368, November 2002.