An Empirical Study of Flooding in Mesh Networks

Thomas Zahn, Greg O'Shea and Antony Rowstron
Microsoft Research, Cambridge, UK

April 2009

Technical Report
MSR-TR-2009-37

Microsoft Research
Microsoft Corporation
One Microsoft Way
Redmond, WA 98052
http://www.research.microsoft.com

ABSTRACT. Flooding in wireless mesh networks is a fundamental operation to many network-level and application-level protocols. Therefore, efficient flooding is important. Prior work has shown that naive flooding can generate broadcast storms. This has inspired much research on optimized flooding, most of which has been based on analysis and simulation.

In this paper, we measure the performance of flooding protocols on a large-scale office 802.11a mesh network of 110 nodes distributed across four floors. We compare three protocols: naive flooding and two optimized flooding protocols. In naive flooding, all nodes rebroadcast received flood messages once. The other two protocols are inspired by the multi-point relay algorithm, which selects subsets of the nodes to rebroadcast.

To understand the scalability of each protocol, we examine its performance with and without background traffic. For the flood protocols, we measure the delivery ratios and packet overhead. All perform well without background traffic. However, contrary to common opinion, with background traffic the optimized flooding protocols perform sufficiently poorly to seriously question their viability in 802.11-based mesh networks. Thus, as a different point in the design space, we also compare implementing discovery, often implemented using flooding, using key-based routing which does not rely on flooding.

1. INTRODUCTION

Advances in wireless networking mean that larger-scale wireless mesh networks are becoming increasingly feasible [14]. Flood-based protocols are widely used in wireless mesh networks at all layers of the stack. Flooding is often used to perform discovery, and in general flooding-based protocols are well suited to the dynamic link-level topology observed in mesh networks.

At the routing-level, many wireless routing protocols, such as LQSR [11], AODV [24] and DSR [18], rely on flooding to perform route discovery. Other wireless routing protocols, such as OLSR [6] rely on flooding to propagate topology information. At the network-level, zero configuration services use discovery to detect IP address conflicts [4], and classic centralized network-services like DNS [21] and DHCP [13] can be implemented without a server [4, 5] in a wireless mesh using flooding. Also at the network-level, Address Resolution Protocol (ARP) implementations, for routing protocols implemented at layer 2.5, use flooding to discover the IP to MAC address mappings for nodes [25]. Application-level services use flood-based protocols, for example, to implement Web Services Dynamic Discovery (WS-Discovery) [1], a standardized XML-based protocol providing generic discovery for applications. WS-Discovery can be used for simple device discovery, such as finding a printer on the network, or in more demanding applications, such as distributed cooperative caches. Flooding is, thus, a fundamental operation for wireless mesh networks. Understanding the behavior of flooding in real-world mesh networks is essential if it is to support existing protocols at scale.

The goal of this paper is to quantitatively evaluate the performance of three broadcast-based flood protocols for wireless mesh networks. Although flooding protocols have been studied through analysis and simulation [23, 30, 29], it has been shown that broadcasts can behave very differently in real-world wireless networks [19]. Nonetheless, wireless routing protocols such as the IETF standardized OLSR rely on optimized flooding that has largely been evaluated through analysis and simulations [26, 20, 2]. In order to understand the real-world performance and limitations, we compare and examine the performance of the three flood protocols using a 110-node 802.11a wireless mesh network.

Throughout this paper, we will refer to broadcasting as the local transmission of a packet by one node to its 1-hop physical topological neighbors and to flooding as the process of propagating a packet throughout the entire mesh network by means of local broadcasts. Naive flooding, with every node simply rebroadcasting every received packet, is simple and robust, but it incurs a high message overhead. This negative impact of flooding has been studied analytically and in simulation [23], and is referred to as the broadcast storm problem. There has been considerable research on how to reduce the message overhead, for example by having only a subset of the nodes rebroadcast, as in multi-point relays (MPRs) [26] in OLSR.

1.1 Contributions

In this paper, we quantitatively evaluate the performance of three different flood-based protocols: naive flooding and two-variants of optimized flooding. The two variants of optimized flooding are MPR-inspired [26] protocols, where only a subset of the nodes are selected to rebroadcast. One is closely based on MPR as used in OLSR [6], the second uses global knowledge to select the smallest subset of nodes to rebroadcast. The second approximates a lower-bound on the overhead achievable by any protocol that attempts to select subsets of nodes.

In order to investigate the performance of the three flood-based protocols, we run a number of experiments on a 110-node 802.11a office wireless mesh testbed. We measure the base performance of each protocol without background traffic. We also measure the performance with background traffic, in the form of a TCP flow. We measure the delivery ratio and message overhead (duplicate packets received) for each protocol, and the impact that each protocol has on the TCP flow. The results show that, as would be expected, the naive flooding provides high delivery ratios but also has the highest impact on the competing TCP flow. However, the results demonstrate that, while the optimized flooding protocols have far less impact on the TCP flow, they can achieve surprisingly low delivery ratios.

In particular, the most optimized flooding algorithm, which selects the smallest subset of nodes to rebroadcast, consistently achieves poor delivery ratios. The other optimized flooding algorithm, most closely based on MPR as used in OLSR, provides good delivery ratios on the full 110-node testbed, but achieves poorer delivery ratios when run on smaller subsets of the main testbed.

The results for the flood-based protocols question the feasibility of using the current generation of optimized flood protocols in 802.11 based mesh networks, and we believe they should influence the future design of such protocols. In particular, there is a clear trade-off for flood-based protocols between efficiency and reliability, and it is unclear how to create distributed protocols that can dynamically control this trade-off.

We therefore also explore a conceptually different point in the design space: key-based routing. Although originally designed for the Internet in the context of Distributed Hash Tables (DHTs) [32, 28, 27], recent work has demonstrated that it is possible to efficiently implement key-based routing for wireless networks [17, 3, 8, 31]. Key-based routing can be used to implement discovery, often implemented using flooding. However, doing so introduces a different trade-off. Flooding protocols are simple and generic, often allowing services to be easily supported. Using key-based routing, on the other hand, often requires the service to be redesigned.

In the next section, we describe the protocols used in the paper. Section 3 describes the experimental setup and the 110-node 802.11a wireless testbed on which the experiments are run. In Section 4, we present the results for the three flood-based protocols as well as for key-based routing. Section 5 summarizes the main insights and takeaways of our study. Section 6 presents related work and provides some discussion about the context of the work. Finally, Section 7 concludes.

2. FLOODING PROTOCOLS

We consider three flooding protocols: *naive flooding*, *part-optimized flooding* and *optimized flooding*. We also consider *key-based routing* as an alternative for when flooding is used for discovery. We now describe each of the protocols used in more detail.

2.1 Naive flooding

Naive flooding is the simplest of the three protocols. A source node initiates a flood by creating a packet, which includes a unique identifier, and broadcasts the packet using a standard 802.11a broadcast frame at 6 Mbps. When a node receives a broadcast packet, using the unique identifier, it checks if the packet has already been received, in which case the node drops this duplicate. Otherwise, this is the first time the node has seen the packet, so the node records the associated unique identifier and schedules a local rebroadcast of the packet, with a delay selected uniformly at random from the range 0 to 10 ms. This jitter ensures that self-interference from other nodes rebroadcasting the flood packet should be low.

This is conceptually simple, with all nodes rebroadcasting each packet at most once. A small amount of short term state is maintained to log the packet identifiers seen. This is similar to the mechanism used in some state-of-the-art mesh routing implementations, for example the Mesh Connectivity Layer (MCL) toolkit from Microsoft Research [11].

2.2 Part-optimized flooding

In naive flooding, each node can receive a copy of the flood packet from each one-hop neighbor. This is clearly inefficient, and there has been considerable research on developing optimizations to reduce this message overhead. We evaluate an optimized flooding protocol inspired by the *Multi-Point Relays* (MPRs) [26] protocol used in OLSR [6]. Rather than having all nodes rebroadcast every flood message, a subset of the nodes are selected to rebroadcast. In MPR, this is achieved by each node selecting a subset of its one-hop neighbors, referred to as the *relay set*, that provide complete coverage of the node's two-hop neighbors. Only the relays of a node will rebroadcast flood messages received from that node. When a flood is initiated by a source node, there are a set of nodes that rebroadcast the flood across the mesh network, and we refer to this set as the *union relay set* for that source node. For each source, there exists a specific union relay set.

For our experiments, we generated a union relay set using the MPR algorithm. To achieve this on our testbed, we obtain measurements of the link quality [7, 12] between nodes in the network from which, after applying a link quality threshold (6 Mbps bidirectional link throughput) to exclude fragile links, we determine the one and two-hop neighbors for each node. In our experiments, we calculate the single union relay set assuming a high-degree node is the source and using OLSR's MPR selection algorithm (as specified in RFC 3626). All members of this union set are then configured to rebroadcast any flood messages that they receive *from any* source. Configuring the union relay set statically aids us in performing repeatable experiments in which the set of nodes rebroadcasting each flood message is constant. We refer to this as *part-optimized flooding*.

2.3 Optimized flooding

The part-optimized flooding protocols uses union relay sets which are potentially not optimal, having more members than strictly required to ensure a given flood message can reach all nodes. This is because MPR is designed to use only information available locally to each node. We have the advantage of being able to use global information.

In the optimized protocol, we generate an approximation to the globally smallest set of nodes that have to rebroadcast a flood packet. To do this, we compute the union relay set for *each* node in the network based on OLSR's MPR selection algorithm. A heuristic driven brute-force search is used to discover redundant entries in each union relay set. This process exploits global knowledge, and generates subsets of each union relay set. From this set, we select the union relay set with the lowest cardinality, which is a close approximation to the smallest union relay set required to provide complete coverage of the mesh, assuming lossless delivery over the thresholded links. The members of this set are configured to rebroadcast each flood packet once. The cardinality of this set was approximately 1/3 of that of the part-optimized union relay set.

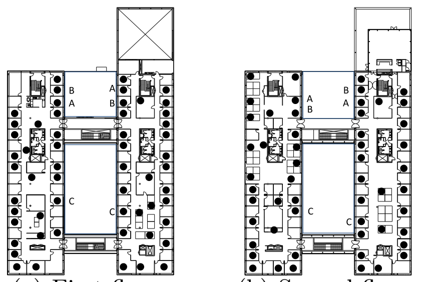
Statically computing the smallest union relay set, while computationally intensive, has the advantage that we achieve close to the minimum number of nodes required to rebroadcast, thereby minimizing the communication overhead. This is important, as there are many proposals for optimized flooding protocols that select a subset of the nodes to rebroadcast. All these protocols have to select sufficient nodes to ensure complete coverage. The union relay set we generate represents a close approximation of the lower bound for any of these protocols. We refer to this as *optimized flooding*.

2.4 Key-based routing

Protocols that require a discovery operation can often also be implemented using key-based routing, which represents a different design point from using flood protocols.

Key-based routing implements a distributed hash table (DHT), where the values are assigned keys and <key, value> pairs are stored in the DHT. Each key is dynamically mapped to a specific node in the mesh network, and a source node can route to a particular key, with the message being delivered to the node responsible for the key. To perform a lookup operation, a node generates the key associated with the value required, and the lookup is routed using that key. Contrary to flooding protocols, in key-based routing all packets are unicast.

In the Internet, DHTs are traditionally implemented as overlays. However, simply running these using a standard wireless routing protocol is not efficient [9, 31]. There have been a number of proposals for efficient DHT implementations for wireless networks that run at the link layer [17, 3, 8, 31]. In this paper, we use the Virtual Ring Routing protocol (VRR) [3] which provides both efficient point-to-point routing and key-based routing. Interestingly, VRR does not use flooding (unlike many other routing protocols). A detailed description of VRR can be found in [3].



(a) First floor (b) Second floor

Figure 1: Office building wireless mesh testbed used for experiments.

It should be noted that the flood-based protocols can support arbitrarily complex queries. In contrast, the key-based routing requires a specific key to be generated. Some applications and services need the full flexibility of complex queries, other applications can use key-based lookups, for example DNS, DHCP, ARP and web caches. In key-based routing, a single node is responsible for each key. If a lookup is performed using a key for which there is no associated value, a negative ACK can be returned. This allows efficient end-to-end detection and recovery from message loss, compared to flood-based protocols.

2.5 Implementation Issues

All three flooding protocols as well as the key-based routing are implemented using the Mesh Connectivity Layer (MCL) toolkit from Microsoft Research [11]. MCL adds a new kernel module that appears as a virtual network adapter to the Windows TCP/IP stack, which allows the use of unmodified IP-based protocols and applications. We have replaced the LQSR routing protocol supplied with MCL by the VRR routing protocol, and used VRR for point-to-point routing as well as key-based routing.

3. EXPERIMENTAL SETUP

We first describe the wireless mesh testbed used in the experiments and, then, the experimental configuration.

3.1 The office mesh testbed

Our experiments were run on a 110-node wireless mesh testbed in an office building. The building is three floors high and consists of two parallel wings separated by an internal atrium and a courtyard. The nodes are PCs distributed around the top two floors of each wing, as shown in Figure 1. The testbed is designed to emulate an office mesh: dedicated mesh nodes are placed

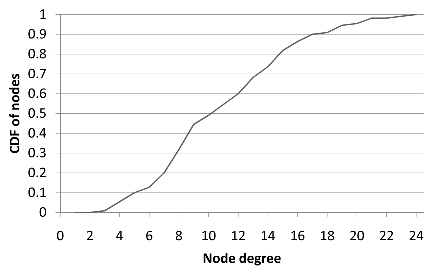


Figure 2: A CDF of node degree.

in offices and open plan office space. We believe the per-floor density and distribution of our machines is a good approximation to what would be achieved in a real deployment.

The majority of nodes are configured with a single 802.11a/b/g adapter card using an Atheros chipset (either a Netgear WAG311 or a D-Link DWL-AG530). We use the Windows XP Atheros reference driver, modified to disable the non-standard Atheros XR and Turbo modes. We operate all cards in ad-hoc mode and choose 802.11a because there are no other 802.11a wireless networks operating in the building. All nodes are also attached to a wired network which we use to issue commands to nodes during experiments and for backhauling results for processing.

In order to minimize interference between floors, each floor operates on a distinct 802.11a channel. The construction and topology of the building prevents effective transmission between the floors through the ceilings, external walls and windows. Transmission between floors, whether vertically between floors on different levels or horizontally across the atrium and courtyard, is achieved by three bridge PCs per floor, each situated adjacent to an external wall. Each bridge PC is configured with two 802.11 cards: one card for communication with nodes on the same floor and the other, whose antenna is mounted on the external face of the wall, for communication with bridge machines on other floors. Three distinct 802.11a channels support inter-floor communication between bridges, and the location of the bridges and the channels assigned to each is marked (with A, B, and C) in Figure 1. Given the topology of our mesh network, if we use a single channel for all inter-bridge links we find that they can become a bottleneck. Partitioning the per-floor bridge machines to use three unique channels minimizes collisions between the inter-bridge links, and thereby increases the throughput of the mesh network.

Figure 2 shows a CDF of node degree for the testbed. A node is considered linked to another when both nodes have detected each other and have each calculated that the Expected Transmission Time [11] on the link between them satisfies a minimum threshold. We can see that while many nodes are well connected, a small number of nodes have few neighbors. These nodes tend to be the nodes near the corners of the floors. High-degree nodes tend to be in the centers of each floor. The diameter of the network is approximately 6 hops.

In its base configuration, our testbed forms a single 110-node mesh. We also use a per-floor configuration where each floor is independent, thereby treating the testbed as four different and distinct mesh networks, each configured with 27 machines using a single wireless interface. The configuration would be representative of the topology if a single internet gateway was added per floor. We believe both the full 110-node configuration and the per-floor configuration represent potential topologies that would be used if a real mesh deployment were done in this building. We also use the multiple topologies to help validate that our results are not an artifact of a particular topology.

3.2 Experimental configuration

We compare the performance of the three flooding protocols and key-based lookup in a number of controlled experiments. The experiments are designed to measure the impact of the protocols on other traffic in the network, and vice versa. In the study, we use the term *lookup* to refer to either the process of performing a single mesh-wide flood for the flooding protocols or a single key-value lookup for key-based routing.

In the base experiment, we use a single TCP flow between two nodes and then, concurrently with this flow, perform a number of lookups using the selected protocol. In order to allow us to compare the protocols, we assume that each lookup is trying to reach a unique node in the mesh. This unique node is selected uniformly at random from the set of nodes in the mesh using a seeded pseudo-random number generator *per lookup* operation. For each set of experiments across the four protocols, we use the same seed for the random number generator. This ensures that across a single experimental run we select the same set of destination nodes for each protocol. In other words, per lookup the destination node for the flooding protocols is always selected to be the same node that would need to receive the message if key-based routing were used. For all protocols a lookup is considered successful if the unique node receives the packet.

We consider two metrics: the TCP flow *throughput* and *delivery ratio*, where delivery ratio is defined as the fraction of successfully delivered lookups. For the flood protocols, we also calculate the coverage ratio, defined as the fraction of nodes that receive each flood-based lookup. As would be expected, in all experiments, the coverage ratio is qualitatively the same as the delivery ratio, so we do not include the results for coverage.

In the experiments, we use the ttcp tool [22] to generate and measure the TCP throughput. For each experiment, we configure ttcp to send 16 MB of data, and we generate lookups at a fixed rate of between 0 and 20 per second throughout the lifetime of the TCP flow. Lookups are issued by randomly selected nodes, again selected using a seeded pseudo-random number generator using the same seed for the same experiment for all four protocols. In order to ensure that there will not be bursts of correlated lookups, lookup initiation is uniformly distributed across each second. None of the protocols use any form of end-to-end retransmission of the lookups.

Each experiment consists of 21 steps, one step for each lookup rate, and each step comprises 10 consecutive ttcp runs at that lookup rate. At the end of each step we verified that the TCP throughput returned to its baseline value, where the *baseline* throughput is defined as the TCP throughput achieved in the *absence* of any lookups. If the baseline throughput changed significantly between steps, which is possible due to environmental issues, we reran the entire experiment. During our initial experiments on the full testbed, we found that link-level load induced by the experiments was causing the paths selected by VRR to flap, as the link-level metrics used by VRR changed due to congestion. Therefore, to remove the effects of path flapping from the results, we altered the VRR driver to ensure that packets between the TCP source and destination would be routed on a fixed path. Therefore, all experiments using the same TCP source and sink route the TCP traffic through the same nodes.

3.3 Mitigating environmental factors

Experiments running on a real large-scale testbed in a standard office building encounter a number of challenges caused by environmental variables. The testbed is susceptible to many factors beyond our control, such as the number of people in the building, their movements and the position of doors, which has also been observed in [10]. These factors impact the quality of links in the network, and would therefore also impact the result of an experiment. In order to mitigate these effects, we ran all the experiments at night or during weekends. We ensured that each single experiment ran to completion in a single pass, without interruption, and incorporated validation-phases before and after each step of every experiment to verify that the baseline performance of the mesh network remained constant. Most experiments required on the order of 2 hours to run, and, for any experiment that failed a verification phase, we reran the entire experiment. We observed some variation between experiments run on different nights. For example, the highest multi-hop throughput TCP flow used in the experiments on the full testbed achieved a baseline throughput of between approximately 900 KB/sec and 1100 KB/sec. In order to remove these effects from the results, we normalize all throughput figures by the baseline throughput measured during each experiment.

4. EXPERIMENTAL RESULTS

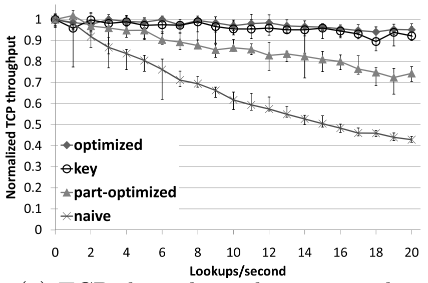
4.1 Base Results

The first set of experiments evaluates the impact of performing lookups concurrently with a TCP-flow on the full 110-node testbed. The TCP-flow source and sink nodes are selected on different floors, such that the path length between them is four hops. While the TCP-flow was active, we selected random nodes to perform lookups. The lookup rate per second was fixed during each step, and we varied the rate from 0 to 20 lookups per second. The experiment was repeated three times, selecting different source and sink nodes for the TCP-flow. We refer to the three TCP-flows used in the experiments as the *high rate*, *medium rate* and *low rate* paths because without any concurrent lookups they achieved a throughput of approximately 1000 KB/sec, 850 KB/sec and 610 KB/sec, respectively. The default packet payload size was 1000 bytes unless stated otherwise.

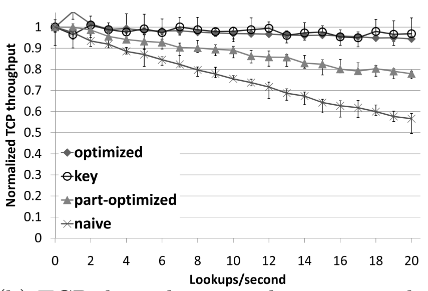
Figures 3(a), (b), and (c) show the normalized median TCP throughput versus the lookup rate for the different protocols for each TCP flow. In each figure, the maximum and minimum TCP throughput achieved during the ten runs is shown using error bars. The results show that key-based routing (*key*) and optimized flooding (*optimized*) have low impact on the TCP throughput, reducing it by only 5-10% at a lookup rate of 20. In key-based routing each lookup traverses a single forwarding path towards the destination key. Optimized flooding uses a union relay set containing only 14 nodes, which rebroadcast packets. Therefore these have low message overhead, explaining why they have low impact on the throughput. The key-based routing also benefits from using unicast transmission for data packets because the 802.11 driver selects an appropriate data rate to transmit the packet, so the spectrum is used more efficiently.

In contrast to optimized flooding, part-optimized flooding (*part-optimized*) uses a union relay set of 38 nodes. This is approximately 2.5 times larger and represents slightly more than one third of the nodes in the testbed. The TCP throughput results in Figure 3 show that part-optimized flooding reduces throughput by 25% at a lookup rate of 20 per second. Indeed, a 10% throughput decrease is observed at only approximately 6 lookups per second. Naive flooding (*naive*) has the highest impact on TCP, as would be expected as it incurs the highest messaging overhead: TCP throughput decreases by a factor of two at 20 lookups per second, and by approximately 10% with just three lookups per second.

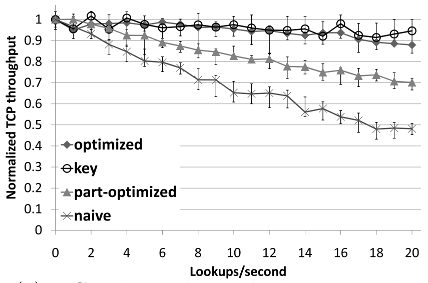
Figures 3(d), (e), and (f) show the delivery ratio versus the lookup rate. All the delivery ratios are inde-



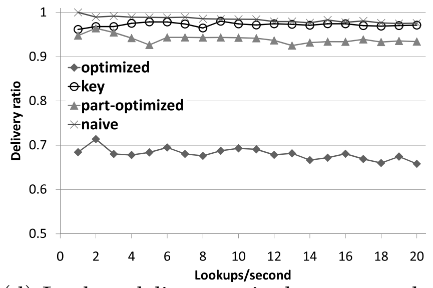
(a) TCP throughput, low rate path



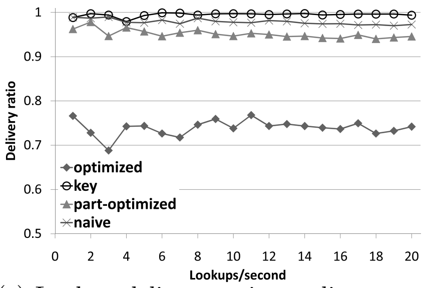
(b) TCP throughput, medium rate path



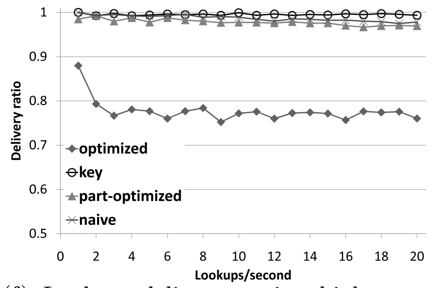
(c) TCP throughput, high rate path



(d) Lookup delivery ratio, low rate path



(e) Lookup delivery ratio, medium rate path



(f) Lookup delivery ratio, high rate path

Figure 3: TCP throughput and lookup delivery ratio against lookup rate on the 110 node testbed.

pendent of the lookup rate, implying that interference from concurrent lookups is small. Key-based lookup achieves the highest delivery ratio, benefitting from its low per-lookup overhead and, unlike the flood-based mechanisms, from using 802.11a unicast transmission for data packets. While there is no end-to-end retransmission of lookups, the 802.11 MAC layer retransmits a unicast frame up to 16 times if an ACK is not received. There is no explicit retransmission of broadcast frames. **Summary:** Naive flooding has the highest impact on the throughput, whereas optimized flooding and key-based routing have low impact. Furthermore, delivery ratios are independent of the lookup rate.

4.1.1 *Naive versus Optimized Flooding*

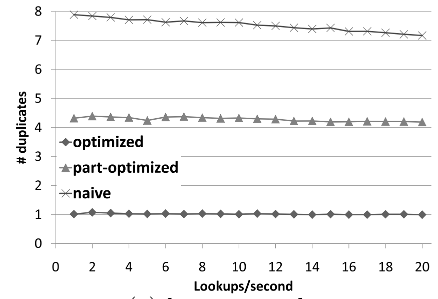
Naive and part-optimized flooding achieve high delivery ratios but, in contrast, optimized flooding performs poorly. All the flood-based approaches achieve resilience to packet loss through redundancy rather than explicit retransmissions: lowering redundancy reduces the impact on TCP throughput, but increases the probability of an unsuccessful lookup. To explore this in more detail, we now look at the number of duplicates received by nodes with the flood-based mechanisms.

Figure 4 shows the mean number of duplicates received per node per lookup versus the lookup rate. The impact of increasing the lookup rate on the number of duplicates received is low for all flood-based mechanisms. This supports the argument that interference from concurrent lookups is low. Naive flooding generates the highest number of duplicates, and this level of redundancy ensures that, even with packet loss, a high delivery ratio is achieved. Nonetheless, naive flooding fails to achieve a delivery ratio of 100%, despite this level of redundancy. In all cases of the naive flooding, if a one-hop neighbor of the lookup source received the lookup, it was delivered successfully. The failed lookups were not received by any neighbors of the source.

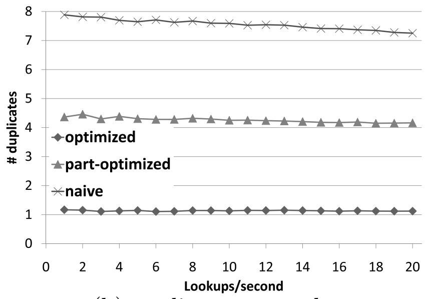
The results for optimized flooding show a much lower level of redundancy. The per-lookup message overheads are lower, which reduces the impact on TCP throughput, but each node only receives the message twice on average. Clearly, from the delivery ratio results, this is insufficient redundancy to ensure successful lookup delivery. In contrast, the part-optimized flooding results show a level of redundancy between that of naive and optimized. Interestingly, this level of redundancy achieves a good delivery ratio, with lower impact on the TCP throughput, compared to naive flooding. However, as we will see in the per-floor testbed results, the level of redundancy achieved by part-optimized flooding *is not always* sufficient to achieve good delivery ratios. **Summary:** Flooding protocols need a certain level of redundancy to ensure high delivery rations (naive and part-optimized flooding achieve high delivery ratios, optimized flooding performs poorly). The lookup rate has little impact on the number of duplicates received. For naive flooding, the critical propagation link is the first hop.

4.1.2 *Lookup Self-Interference*

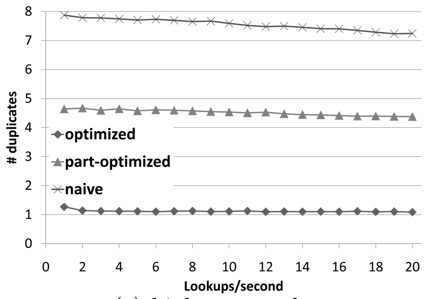
To quantify the impact of lookup self-interference versus the impact from the TCP-flow, we ran an experiment without a competing TCP-flow. For this experi-



(a) low rate path

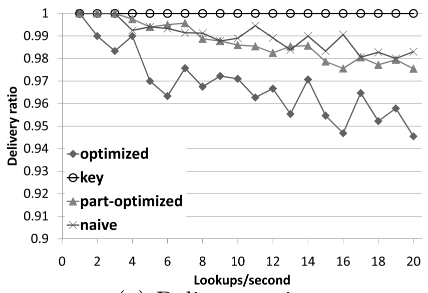


(b) medium rate path

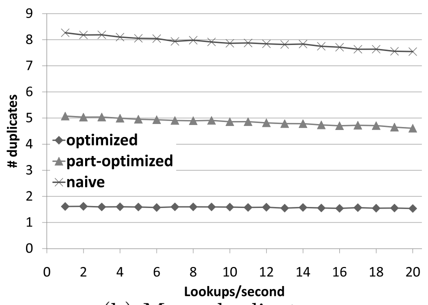


(c) high rate path

Figure 4: Mean duplicates received per node per lookup on the 110 node testbed.



(a) Delivery ratio



(b) Mean duplicates

Figure 5: Lookup delivery ratio and mean duplicates per lookup without any TCP flow.

ment, each lookup rate was sustained for 100 seconds. Figure 5 shows the delivery ratio and the mean number of duplicates per node against lookup rate. Note the tighter scale on the y-axis in Figure 5(a). At one lookup/second all the protocols achieve a delivery rate of 100%, implying that the subset of nodes selected to rebroadcast in the optimized and part-optimized flood protocols provide sufficient coverage to be able to deliver the flood message. The key-based lookup achieves 100% delivery ratio, independent of the lookup rate. The flood-based mechanisms show that increasing the lookup-rate incurs a small impact in delivery ratio, implying that there is some self-interference, which can also be observed in the small decrease in the number of duplicates received. More interestingly, without any competing TCP-flow, optimized flooding attains a lookup delivery ratio of 95% and above. This should be compared with the results in Figures 3(d), (e), and (f), where the optimized flood achieves a delivery ratio of approximately only 75%. This shows that a *single* TCP-flow has significant impact on the delivery ratio for the optimized flood.

Summary: These results suggest that self-interference is a minor issue, and that if the level of redundancy is too low, competing traffic can seriously impact successful delivery of lookups.

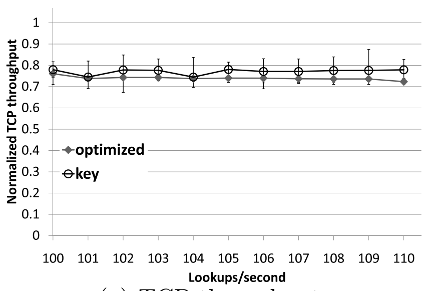
4.1.3 *High Lookup Rates*

Finally, we ran some experiments at higher lookup rates, varying the rate from 100 to 110 lookups per second, so that, at a rate of 110, each node issued one random lookup per second. Figure 6(a) shows the TCP throughput for the medium rate path versus lookup rates for both key-based routing and optimized flooding. At these high rates there is a decrease in the TCP throughput of just over 20% for both key-based routing and optimized flooding. In contrast, at a lookup rate of only 20 per second, naive flooding decreases the TCP throughput by 50% and part-optimized flooding by approximately 20%. Figure 6(b) shows the delivery ratios, with key-based routing achieving high delivery ratios. However, as seen in previous experiments, optimized flooding achieves low delivery ratios. We also ran the same experiment for naive and part-optimized flooding, but these high lookup rates could not be supported and the TCP-flow collapsed.

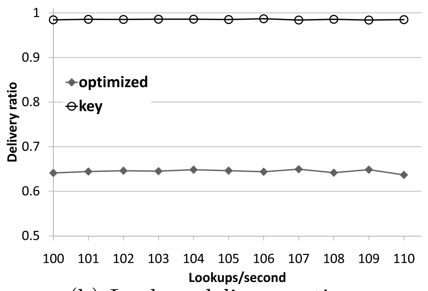
Summary: Naive and part-optimized flooding do not cope well with high lookup rates.

4.2 Payload size sensitivity

Next, we examine the sensitivity to the lookup payload size. To this end, we reran the medium path rate experiments using lookup payload sizes of 100 and 500 bytes. Figure 7 shows the TCP throughput achieved versus lookup rate, with lookup payload sizes of 100 and 500 bytes. There is no impact on TCP throughput from key-based lookup as the payload size is varied. In contrast, payload size affects TCP throughput for naive flooding. In Figure 3(c), we observed approximately a 50% reduction in TCP throughput at 20 lookups per

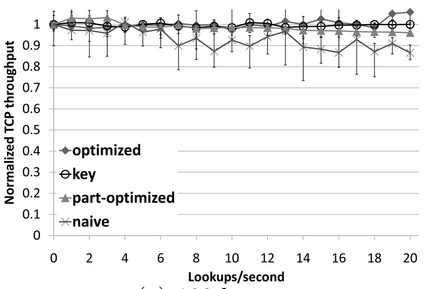


(a) TCP throughput

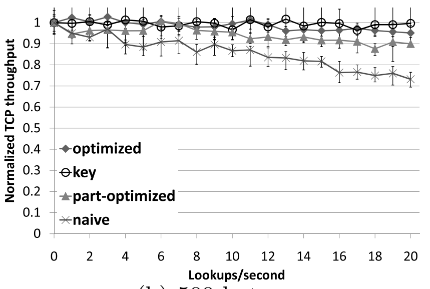


(b) Lookup delivery ratios

Figure 6: High lookup rates with the medium rate path.



(a) 100 bytes



(b) 500 bytes

Figure 7: TCP throughput of lookups with payload sizes of 100 and 500 bytes on the medium rate path.

second when using a payload size of 1000 bytes, decreasing to 25% at 500 bytes and 8.5% at 100 bytes. As the payload size becomes small, the per-packet fixed overheads (802.11 headers etc) begin to dominate. This explains why, between 1000 and 500 byes, the impact on the TCP throughput drops by a factor of 2, while from 500 to 100 the impact drops by a factor of 3 rather than by a factor of 5, as may be expected. Both the optimized and part-optimized flooding also have less impact on the TCP throughput as the payload size is reduced, but the effect is less pronounced.

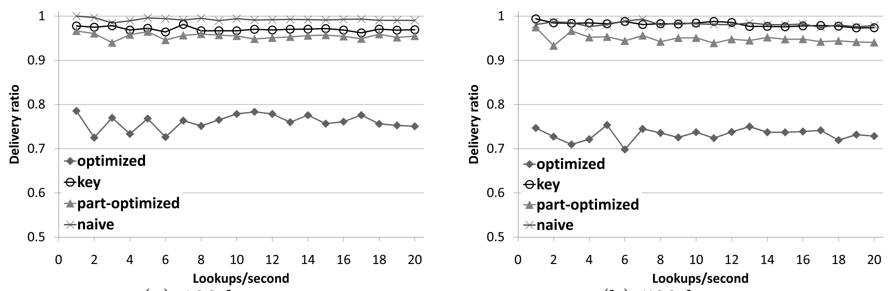
Figure 8 shows the delivery ratio versus the lookup rate for the different payload sizes, showing only marginal differences from the results for 1000 byte payload sizes. This is consistent with our observation that the TCP-flow impacts the delivery ratio more than self-interference.
Summary: Most pronounced in naive flooding, the impact on the TCP throughput correlates with the payload size. This would imply that protocols like WS-Discovery that use XML will have a far more significant impact on mesh networks than protocols that use small lookup packets.

4.2.1 *Naive versus Optimized Flooding*

Figure 9 shows the mean number of duplicates per node per lookup versus lookup rate for 100 and 500 bytes, with Figure 4(b) showing the results for 1000 bytes. The lookup payload size has little impact on the number of duplicates for optimized flooding and a small impact for part-optimized flooding. However, there is considerable impact on the number of duplicates received in naive flooding. The mean number of duplicates received drops from approximately 12.5, at 100 bytes, to 7.5 at 1000 bytes.

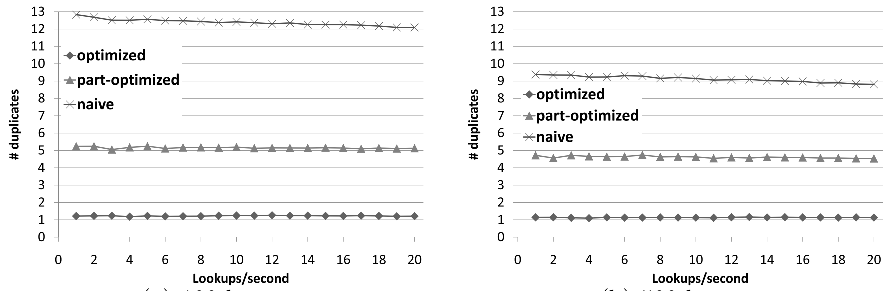
Recall that in the naive flooding, we use a jitter selected uniformly at random from the range [0,10] ms, thus the average jitter duration is 5 ms. The time taken for 802.11a to broadcast a 1000 bytes packet (at 6 Mbps) will be in the order of 1.5ms compared to only 0.3 ms for a 100 byte packet. From Figure 3, we can see that the median node degree is approximately 10. Hence, with an average jitter of 5ms, there is significant potential to incur local self-interference between neighbors when rebroadcasting the larger packets for the same lookup.

To investigate further, we ran an experiment using naive flooding with no competing TCP flow, a jitter selected uniformly from the range [0,50] ms and 1000 byte lookup packets. We observed that the number of duplicates received increased by approximately 50% for naive flooding; comparable to the number of duplicates for 100 byte packets. Increasing the mean jitter to 25ms reduces the likelihood of self-interference for the larger packets. However, this also increases the end-to-end delivery latency and, as naive flooding has high delivery ratios in all experiments, we therefore used [0,10] ms jitter in the experiments. For optimized and part-optimized flooding, there is little effect: only a small subset of the nodes rebroadcast lookups, and the lower density of broadcasts in the network reduces the likelihood of local self-interference. This explains why we see



(a) 100 bytes (b) 500 bytes

Figure 8: Delivery ratio of lookups with payload sizes of 100 and 500 bytes on the medium rate path.



(a) 100 bytes (b) 500 bytes

Figure 9: Mean duplicates received per node per lookup with payloads of 100 and 500 bytes on the medium rate path.

little impact of payload size on the number of duplicates for optimized and part-optimized flooding.

Summary: The lookup payload size has little impact on the number of duplicates received for optimized flooding, a small impact for part-optimized flooding, and considerable impact in naive flooding.

4.3 Per-floor testbed results

The results presented so far use the full 110-node testbed, and next we examine the performance using the four smaller 27-node testbeds. We used the same set of experiments as in the previous section, selecting a TCP flow of just one hop in length, and fixing the lookup packet size at 1000 bytes.

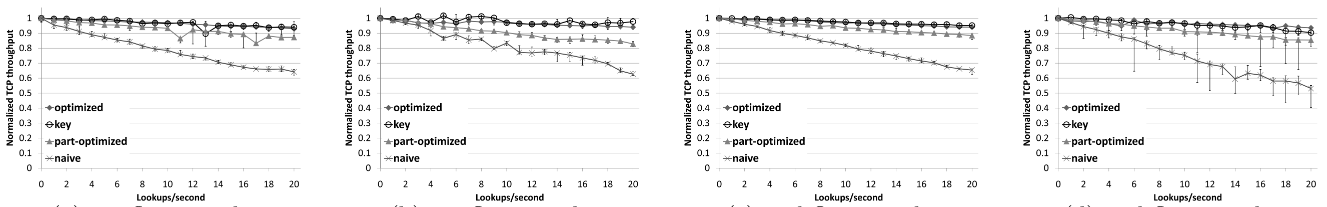
Figure 10 shows the TCP throughput versus lookup rate for each of the four small testbeds. For naive flooding, as with the 110-node testbed, the TCP throughput drops as the lookup rate increases. However, the lookups have less impact on TCP throughput than in the 110-node testbed. This is because the full testbed uses four-hop routes, whereas the smaller testbeds use single hop routes. In order to verify this, we ran an experiment on one of the smaller testbeds, using a two-hop TCP flow and observed that the impact on the TCP flow throughput increased. The more hops that a data packet has to traverse, the more potential there is for the packet to either be delayed or lost, either of which will reduce throughput.

Figure 10 also shows the results for optimized and part-optimized flooding. For optimized, the size of the union relay set was 2 for all floors except for the first floor south where it was 3. For part-optimized, the union relay set size for the first floor north and first floor south was 5 and 10, respectively, while it was 6 for both the second floor north and second floor south. As with the naive flooding, and for the same reasons, there was less impact on the TCP throughputs in the smaller testbeds compared to the full 110-node testbed. Across all four testbeds, optimized outperformed part-optimized, again with optimized having similar impact to key-based routing. Given the union relay set size of only 2 (or 3 in one testbed) for optimized flooding this is to be expected: the number of transmissions incurred per lookup was similar to key-based routing.

Summary: For naive flooding, the throughput drops with an increasing lookup rate, but the impact is less than in the 110-node testbed. Optimized outperformed part-optimized flooding, having similar impact as key-based routing.

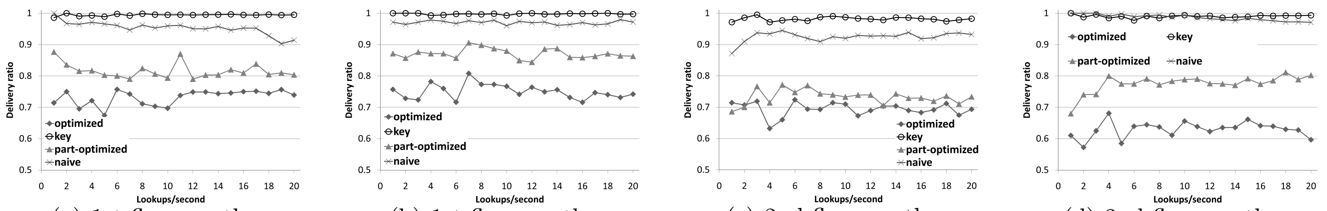
4.3.1 *Naive versus Optimized Flooding*

Figure 11 shows the delivery ratio versus lookup rate on the four small testbeds. Again, key-based routing and naive flooding perform well across all four testbeds. Significantly, in contrast to the results obtained on the full 110-node testbed, both optimized and part-optimized flooding achieve unacceptably poor delivery ratios across all the smaller testbeds. In the full testbed, part-optimized



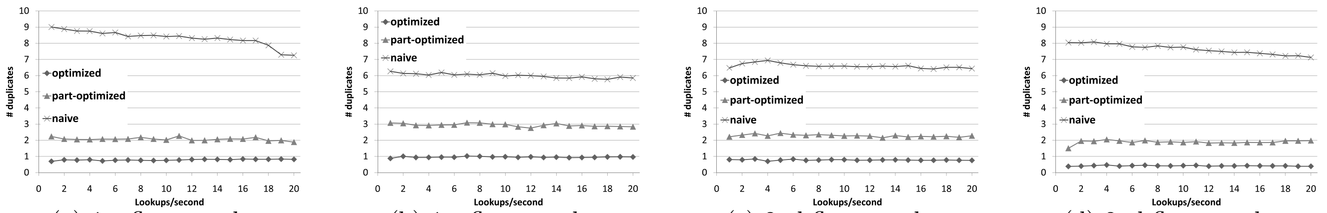
(a) 1st floor north (b) 1st floor south (c) 2nd floor north (d) 2nd floor south

Figure 10: TCP throughput versus lookup rate on the four 27-node testbeds.



(a) 1st floor north (b) 1st floor south (c) 2nd floor north (d) 2nd floor south

Figure 11: Delivery ratio versus lookup rate on the four 27-node testbeds.



(a) 1st floor north (b) 1st floor south (c) 2nd floor north (d) 2nd floor south

Figure 12: Mean duplicates received per node per lookup on the four 27-node testbeds.

achieved a good delivery ratio. As with the full testbed results, we observe that the delivery ratio is independent of the lookup rate, which implies that the poor performance is not due to interference between the lookups, but rather due to the TCP flow. Again, we confirmed this by running the experiment on one floor without the TCP flow. We observed that the delivery ratio for both was again independent of the lookup rate and that it increased to above 97% for all lookup rates.

To understand why the performance of part-optimized flooding is different from that of optimized flooding, we need to look at the number of duplicates received. Figure 12 shows the number of duplicates as a function of lookup rate for the four small testbeds. For naive flooding, we observe the number of duplicates is high which, as for the full testbed, yields a good delivery ratio. For the optimized flood, on average, one or less duplicates per lookup are received by each node, which is similar to the number of duplicates received in the full testbed results (Figure 4). As with the full testbed, this level of redundancy is too low to achieve resilience to packet loss. Figure 12 also shows that for the part-optimized, the number of duplicates is between two and three across the testbeds. This level of redundancy is significantly below the level observed for the full testbed (Figure 4) and not sufficient to sustain high delivery ratios. To explain this, we need to bear in mind that, on the one-hop flow, TCP achieves higher rates as compared to the four-hop paths in the full testbed (approx. 2.4MB/s vs. 1MB/s). Therefore, this TCP flow will cause more medium contention. Furthermore, as we are restricted to one floor, flood lookups are also more likely to be effected by this TCP flow.

To further confirm this, we also investigated whether the TCP flow was causing the lookup packets to be lost at the point of origin, as was happening in the naive flooding in the full testbed. This proved not to be the case. Instead, we observed that virtually all of the lookups were received by several nodes in the network and the unsuccessful deliveries occurred because either a union relay set member did not receive the lookup, and hence could not rebroadcast it, or because the destination node did not receive the lookup from a union relay set member.

Summary: Unlike in the 110-node testbed, both optimized *and* part-optimized achieved poor delivery ratios in all four smaller testbeds.

5. INSIGHTS AND TAKEAWAYS

Our results for flood-based protocols should inform the design of future optimized protocols. We have shown that naive flooding achieves good delivery ratios, but impacts the performance of other traffic in the network. Optimized flooding has little impact on other traffic, but achieves poor delivery ratios. Interestingly, the self-interference from concurrent lookups appears to be generally low.

Furthermore, we observed that for naive flooding, the critical propagation link is the first hop. If a one-hop neighbor of the flood source received the flood packet, it was eventually delivered successfully. Failed floods were not received by any neighbors of the source and, thus, were lost at the source.

Our results illustrate the fundamental problem for flood-based protocols: the trade-off between efficiency and reliability. The results for part-optimized flooding (based on the widely used MPR algorithm) show that, in the large 110-node testbed, it provides a good balance between efficiency and reliability. However, in the smaller testbeds, we observed that it does not.

This implies that protocols explicitly need to manage this trade-off. However, achieving this is difficult. In this study, for the optimized flooding approaches, we performed offline processing to determine the subset of nodes to rebroadcast. A major challenge is that the set of nodes selected to rebroadcast is usually selected and maintained using a distributed algorithm running over the mesh. It seems difficult for such algorithms to know if they are selecting too few nodes to rebroadcast messages. This leads us to question the viability of such approaches. Interestingly, the MPR algorithm is the basis of routing protocols like OLSR, which questions how well OLSR will work in real mesh networks.

Flood-based protocols are often used in wireless mesh networks to implement discovery because of their simplicity and their expressiveness. As illustrated, it remains unclear how to dynamically manage the trade-off between efficiency and reliability. Therefore, we also evaluated the use of key-based routing to perform discovery. The results show that key-based routing performs well, having both low impact on TCP throughput and high delivery ratios across all experiments. However, the use of key-based routing introduces a different trade-off: The price of obtaining such performance is that network-level and application-level services may need to be rearchitectured in order to exploit key-based routing, and supporting arbitrarily complex queries with key-based routing may not always be possible.

6. DISCUSSION

There has been much research into efficiently performing flooding in wireless networks. Most of the work is either analytical or simulation-based [23, 30, 29]. Recently, it has been observed that many of the simplifying assumptions made when modeling or simulating protocols may invalidate the results in real-world deployments [15]. There has been little work on the evaluation of broadcast mechanisms in real testbeds of any significant size. A notable exception is the work on evaluating flood-based protocols in a 150-node sensor network [16]. Sensor networks use low-power, low-bandwidth wireless radios, and the nodes largely perform a single dedicated task. In contrast, in an office mesh using 802.11a, there is higher bandwidth and the mesh is expected to support multiple services. Therefore, in this paper, we examine not only the performance of lookups but also examine the effect of, and impact on, cross-traffic.

Our results evaluate the impact of performing up to 20 lookups per second across the testbed, which is equal to one lookup per node every 5.5 seconds for the full 110-node testbed. This rate is higher than that observed on wired networks today. However, we believe these rates are realistic if mesh networking becomes widely adopted, for example the suggested default interval at which OLSR nodes broadcast Topology Control messages [6] is 5 seconds. Aggregating the rate of floods across all network- and application-level services, a combined lookup rate of 20 per second across a 110-node network seems very feasible.

In the experiments, we used lookup packet sizes that varied by an order of magnitude, from 100 to 1000 bytes. The 100 bytes is representative of the size order that network-level services use today, for example, ARP and IPv6 Neighbor Discovery protocols require 42 and 86 bytes, respectively. The WS-Discovery specification uses XML-encoded messages in SOAP envelopes [1]. The basic example discovery message in the specification is 673 bytes, and if a compact signature is included it adds a further 191 bytes. We therefore believe that a significant fraction of lookup payload sizes in the future will be large.

7. CONCLUSIONS

We have studied three different broadcast-based flooding protocols for use in wireless mesh networks. Flooding protocols are often used to implement discovery, and, in this paper, we have also studied the performance of implementing discovery using key-based routing.

The study shows that the flood-based protocols trade delivery ratio for impact on the competing traffic. The study highlights that optimizing the flood protocol by selecting subsets of the nodes to perform the rebroadcast, as in MPR-based algorithms does reduce the impact on the competing traffic, but at the price of significantly lowering delivery ratios. We have demonstrated the need for flood-based protocols to dynamically control the size of the subset selected, or use other techniques, to control this trade-off. This seems non-trivial if the protocol is to be distributed, scalable and able to adapt to dynamic network conditions. Key-based routing has been shown to perform well, having little impact on a competing TCP flow and achieving high delivery rates. However, using key-based routing often requires that services be rearchitectured.

8. REFERENCES

[1] J. Beatty, G. Kakivaya, D. Kemp, T. Kuehnel, B. Lovering, B. Roe, C. S. John, J. Schlimmer, G. Simonnet, D. Walter, J. Weast, Y. Yarmosh, and P. Yendluri. Web Services Dynamic Discovery (WS-Discovery), Apr. 2005. `http://specs.xmlsoap.org/ws/2005/04/discovery/ws-discovery.pdf`.

[2] A. Busson, N. Mitton, and E. Fleury. Analysis of the multi-point relay selection in olsr and implications. *Challenges in Ad Hoc Networking*, 197, 2006.

[3] M. Caesar, M. Castro, E. Nightingale, G. O'Shea, and A. Rowstron. Virutal Ring Routing: Network routing inspired by DHTs. In *Sigcomm'06*, Sept. 2006.

[4] S. Cheshire, B. Aboba, and E. Guttman. Dynamic Configuration of IPv4 Link-Local Addresses (RFC 3927), May 2005. `http://ietf.org/rfc/rfc3927.txt`.

[5] S. Cheshire and M. Krochmal. Multicast DNS (Internet Draft), June 2005.

[6] T. Clausen and P. Jacquet. OLSR RFC3626, Oct. 2003. `http://ietf.org/rfc/rfc3626.txt`.

[7] D. D. Couto, D. Aguayo, J. Bicket, and R. Morris. A high-throughput path metric for multi-hop wireless routing. In *Mobicom*, 2003.

[8] C. Cramer and T. Fuhrmann. ISPRP: A message-efficient protocol for initializing structured P2P networks. In *24th IPCCC*, 2005.

[9] C. Cramer and T. Fuhrmann. Performance evaluation of Chord in mobile ad hoc networks. In *MobiShare*, 2006.

[10] S. M. Das, H. Pucha, K. Papagiannaki, and Y. C. Hu. Studying wireless routing link metric dynamics. In *ACM IMC 2007*, 2007.

[11] R. Draves, J. Padhye, and B. Zill. Comparison of routing metrics for static multi-hop wireless networks. In *SIGCOMM'04*, Aug. 2004.

[12] R. Draves, J. Padhye, and B. Zill. Routing in multi-radio, multi-hop wireless mesh networks. In *MobiCom'04*, Sept. 2004.

[13] R. Droms. Dynamic Host Configuration (RFC 2131), Mar. 1997. `http://ietf.org/rfc/rfc2131.txt`.

[14] J. Eriksson, S. Agarwal, P. Bahl, and J. Padhye. Feasibility study of mesh networks for all-wireless offices. In *Mobisys*, June 2006.

[15] R. Friedman, D. Gavidia, L. Rodrigues, A. Viana, and S. Voulgaris. Gossiping on MANETs: The beauty and the beast. *ACM Operating Systems Review*, Oct. 2007.

[16] D. Ganesan, B. Krishnamachari, A. Woo, D. Culler, D. Estrin, and S. Wicker. An empirical study of epidemic algorithms in large scale multihop wireless networks, 2002. Intel Research IRB-TR-02-003.

[17] Y. Hu, H. Pucha, and S. Das. Exploiting the synergy between peer-to-peer and mobile ad-hoc networks. In *Hot-OS IX*, May 2003.

[18] D. Johnson and D. Maltz. Dynamic source routing in ad hoc wireless networks. In *Ad Hoc Networking*, 2001. Chapter 5, pg 139-172, Addison-Wesley.

[19] H. Lundgren, E. Nordstrom, and C. Tschudin. The gray zone problem in ieee 802.11b based ad hoc networks. *SIGMOBILE Mob. Comput. Commun. Rev.*, 6(3), 2002.

[20] B. Mans and N. Shrestha. Performance evaluation of approximation algorithms for multipoint relay selection. In *Med-Hoc-Net'04*, 2004.

[21] P. Mockapetris. Domain Names - Implementation and Specification (RFC 1035), Nov. 1987. `http://ietf.org/rfc/rfc1035.txt`.

[22] M. Muuss. The story of the ttcp program. `http://ftp.arl.army.mil/~mike/ttcp.html`.

[23] S. Ni, Y. Tseng, Y. Chen, and J. Sheu. The broadcast storm problem in a mobile ad hoc network. In *MOBICOM*, Aug. 1999.

[24] C. Perkins and E. Royer. Ad hoc on-demand distance vector routing. In *Mobile Computing Systems and Applications*, Feb. 1999.

[25] D. Plummer. Converting Network Protocol Addresses to 48.bit Ethernet Address for Transmission on Ethernet Hardware (RFC 826), Nov. 1982.

[26] A. Qayyum, L. Viennot, and A. Laouiti. Multipoint relaying for flooding broadcast messages in mobile wireless networks. In *HICSS*, Jan. 2002.

[27] A. Rowstron and P. Druschel. Pastry: Scalable, distributed object location and routing for large-scale peer-to-peer systems. In *Middleware*, Nov. 2001.

[28] I. Stoica, R. Morris, D. Karger, M. Kaashoek, and H. Balakrishnan. Chord: A scalable peer-to-peer lookup service for internet applications. In *ACM SIGCOMM'01*, Aug. 2001.

[29] B. Williams and T. Camp. Comparison of broadcasting techniques for mobile ad hoc networks. In *MobiHoc '02*, 2002.

[30] Y. Yi, M. Gerla, and T. J. Kwon. Efficient flooding in ad hoc networks: A comparative performance study. In *IEEE ICC '03*, May 2003.

[31] T. Zahn and J. Schiller. MADPastry: A DHT substrate for practicably sized MANETs. In *ASWN*, June 2005.

[32] B. Zhao, J. Kubiatowicz, and A. Joseph. Tapestry: an infrastructure for fault-resilient wide-area location and routing. In *TR UCB//CSD-01-1141, U.C. Berkeley*, Apr. 2001.