MISTRAL: Efficient Flooding in Mobile Ad-hoc Networks*

Stefan Pleisch† Mahesh Balakrishnan‡ Ken Birman‡ Robbert van Renesse‡

†Swiss Federal Institute of Technology (EPFL)
CH-1015 Lausanne, Switzerland

‡Department of Computer Science
Cornell University, Ithaca, NY 14853, USA

stefan.pleisch@epfl.ch {mahesh|ken|rvr}@cs.cornell.edu

ABSTRACT

Flooding is an important communication primitive in mobile ad-hoc networks and also serves as a building block for more complex protocols such as routing protocols. In this paper, we propose a novel approach to flooding, which relies on proactive compensation packets periodically broadcast by every node. The compensation packets are constructed from dropped data packets, based on techniques borrowed from forward error correction. Since our approach does not rely on proactive neighbor discovery and network overlays it is resilient to mobility.

We evaluate the implementation of Mistral through simulation and compare its performance and overhead to purely probabilistic flooding. Our results show that Mistral achieves a significantly higher node coverage with comparable overhead.

Categories and Subject Descriptors

C.2.1 [**Computer-Communication Networks**]: Network Architecture and Design—*Wireless communication*;
C.4 [**Computer-Communication Networks**]: Performance of Systems—*Reliability, availability, and serviceability*;
C.4 [**Computer-Communication Networks**]: Performance of Systems—*Fault tolerance*

General Terms

Algorithms, reliability

Keywords

Mobile ad hoc networks, MANET, flooding, forward error correction, compensation

*Our effort is supported by the Swiss National Science Foundation (SNF), NSF Trust STC, the NSP NetNOSS program, and the DARPA ACERT program.

MobiHoc'06, May 22–25, 2006, Florence, Italy.

1. INTRODUCTION

Mobile ad hoc networks (MANETs) have received much attention in recent years. A MANET is a multi-hop wireless network without fixed infrastructure, in which nodes can be mobile. MANETs are increasingly important because wireless communication is rapidly becoming ubiquitous. Potential applications range from military and disaster response applications to more traditional urban problems such as finding desired products or services in a city. The devices themselves are diverse, including PDAs, cell phones, sensors, laptops, etc. Many new protocols have been proposed to solve the technical problems confronted in MANETs and to offer platform support for applications that collect and exploit the data available in such settings.

Because of the lack of a fixed communication infrastructure, flooding in MANETs [10] is an important communication primitive and also serves as a building block for more complex protocols such as AODV [21] or ODMRP [16]. *Flooding* is the mechanism by which a node, receiving flooded message m for the first time, rebroadcasts m once. We distinguish between flooding and *broadcast*, which is a transmission that is received by all nodes within transmission range of the broadcasting node. Flooding usually covers all the nodes in a network, but can also be limitted to a set of nodes that is defined by a geographical area (also called *geocast flooding* [14]) or by the time-to-live (TTL) parameter of m. Thus, a node receiving the flooded message only rebroadcasts it if it is within the specified area or if the message's TTL is greater than 0.

Unfortunately, flooding has been shown to be susceptible to contention even in reasonably dense networks [18]. Indeed, flooding leads to a large amount of redundant messages that consume scarce resources such as bandwidth and power and cause contention, collisions and thus additional packet loss. Every node receives the message from every neighbor within transmission range, except when messages are lost due to contention and collisions. This problem is known as the *broadcast storm* problem [18]. Because flooding is important in MANET applications, there is a clear need for storm-resistant flooding protocols that operate efficiently. However, reducing the number of redundant broadcasts leads to a lower degree of reliability. Hence, the challenge we face is to strike a balance between message overhead (i.e., the level of redundancy) and reliability.

To reduce the number of redundant messages, two ba-

sic classes of mechanisms have been proposed: (1) imposing a (partial) routing overlay structure; and (2) selectively dropping messages. Approaches in (1) build and maintain a (partial) routing overlay structure in the ad hoc network, which is used to efficiently broadcast the flooded message. For instance, only nodes that are part of a multicast tree rebroadcast the message [20]. Other approaches in this category are [3, 8, 17]. With mobile nodes the underlying routing structure needs to be frequently changed, incurring high maintenance costs and generally reduced reliability during the restructuring. In contrast, approaches in (2) do not rely on an explicit underlying routing structure. Instead, each node uses local information to make an independent decision whether to rebroadcast or to drop the flooded message. The simplest approach in this class is purely probabilistic flooding [18], in which messages are rebroadcast with a certain fixed probability. While probabilistic flooding reduces the number of broadcasts, when applied naively it simply recreates our earlier problem: poorly connected nodes (those with few neighbors) may fail to receive a flooded message. This consideration has motivated a number of more complex approaches, such as the algorithms given in [28, 24].

In our paper, we focus on class (2) but propose a new mechanism to reduce the number of missed flooded messages. We start with purely probabilistic flooding [18] but compensate for dropped data packets by periodically broadcasting *compensation packets*. Every compensation packet encodes a set of packets that have been dropped (i.e., that are not rebroadcast) by the sender. A node's neighbors, upon receipt of such a packet, can recover missing packets if it already has received and buffered a sufficient percentage of the packets that were used in constructing the compensation packet.

Even when a node has lost too many packets to reconstruct missing data, the compensation packets provide information that can be used to identify the loss. We include a secondary recovery mechanism that kicks in when a node discovers an unrecoverable loss, and part of our task in the evaluation presented here is to quantify the tradeoff between the additional message overhead versus increased reliability.

We have implemented Mistral and simulate its performance on JiST/SWANS, a simulation package that lets the developer run code in an emulated environment. Our results show that compensation packets significantly increase coverage when compared to probabilistic flooding with comparable overhead.

The remainder of the paper is structured as follows: Section 2 overviews the problem of flooding and places our work in the context of earlier work. In Section 3 we introduce the Mistral algorithm. Section 4 provides a simple analysis of Mistral. In Section 5, we present the simluation results and measure Mistral's performance. We conclude the paper with Section 6.

2. FLOODING IN MANETS

In any flooding mechanism, one must balance reliability against message overhead. On the one hand, increasing reliability generally involves sending a greater number of redundant messages and thus incurs a higher message overhead. In this worst case, the system risks provoking broadcast storms. Yet redundant messages are needed to reach all nodes and to recover from packet loss, hence reducing the overhead will generally decrease reliability.

The broadcast storm problem is so common in flooding algorithms that it has engendered a whole area of research. Storm-sensitive flooding approaches can be broadly classified into two classes: *local-knowledge-based* and *overlay-based*. Local-knowledge-based approaches decide on whether to rebroadcast or drop a flooded message solely on the basis of local information. Most commonly, they use information from received broadcasts to adaptively determine the forwarding policy. Such algorithms are a natural fit for MANETs, as they do not need to maintain any kind of complex node-to-node state that might need to be adapted in the event of mobility or other topology changes. In contrast, overlay-based approaches structure the node field according to some (local) topology, and then use topological information to efficiently implement flooding and reliability. The problem here is that if nodes have low quality connections to neighbors and/or are in motion, the overlay structure must be adapted. As a consequence, a high rate of management messages may be required, and if a flooded message is propagated while the overlay is out of date, that message may experience a high loss rate. In the worst case, the system might end up in a state of churn, constantly adapting the overlay but never managing to achieve the high quality of flooding that the overlay is intended to support.

We now briefly overview existing work and assign it to the corresponding class. For reasons of brevity, our review is deliberately partial; we focus on results that inspired our work here, or that have been widely cited in the literature. For a more comprehensive overview that includes a comparison of some of the major flooding approaches the reader is referred to [26].

2.1 Overlay-Based Approaches

As just indicated, we use the term *overlay* very broadly. For us, an overlay-based approaches is an algorithm that superimposes a routing structure onto the ad hoc network in support of flooding and rebroadcast. Depending on the position of a node in this overlay, it decides to either rebroadcast a flooded packet, or to only process and then drop it. While overlays provide a convenient mechanism to reduce the message overhead of flooding and to increase reliability, they suffer from the need to reconfigure the overlay when connectivity changes or if the nodes are mobile. Restructuring adds overhead but also increases the likelihood that messages will be lost, and thus may decrease coverage of the flooding protocol.

Ni *et al.* [18] propose to structure the nodes into clusters. Their solution rebroadcasts a packet in a manner that depends on the node's position in the cluster: only cluster head and gateway nodes rebroadcast.

In [8], the goal is to provide low-latency flooding. This is in part achieved by minimizing the collisions and interference. Gandi *et al.* show that an optimal solution to this problem is NP complete, instead, they propose an approximation algorithm. They construct a multicast tree and compute a rebroadcasting schedule such that the expected rate of collisions will be low.

Other approaches are based on the approximation of (minimal) connected dominating sets (MCDS), e.g., [5] [3]. Informally, a dominating set (DS) contains a subset of all nodes such that every node not in the DS is adjacent to one in the DS. Thus, a DS creates a virtual backbone that can be used to efficiently flood messages. It has been shown that

the creation of an MCDS is NP-complete. Thus, most approaches attempt to find a sufficiently good approximation to a MCDS.

A number of approaches rely on two-hop neighbor information to select nodes that rebroadcast the message. These approaches require that *hello* messages containing neighbor information are exchanged between the nodes.

For instance, in the Double-Covered Broadcast (DCB) [17], node n collects information about the two-hop neighbor set. Among its one-hop neighbors it then picks nodes that rebroadcast the message (called *forward node*) such that (1) the rebroadcast by the forward node covers the two-hop neighbors, and (2) the one-hop neighbors that are no forward nodes are within range of at least two rebroadcasts by forward nodes. The reception of the message by the forward node is implicitly acknowledged when n overhears the rebroadcast.

The scalable broadcast algorithm (SBA) [20] also uses two-hop neighbor knowledge, but employs a different approach to select the forward nodes.

With node mobility, the two-hop neighbor sets need to be updated frequently. Otherwise, the neighbor sets become outdated and reliability drops (as observed in [17]).

2.2 Local-Knowledge-Based Approaches

Local-knowledge-based approaches generally decide on a per-node basis whether to rebroadcast a particular flooded message. In the simplest case, each node flips a coin and rebroadcasts messages with a certain probability p [18]. We call this approach *purely probabilistic flooding (PPF)*.

There are a number of variants on this basic idea. For example, one set of algorithms base the rebroadcast decision either on the number of already overheard rebroadcasts, or on the distance or location of the overheard rebroadcast's sender [18]. The idea underlying these schemes is that the additional coverage gained by rebroadcasting decreases with the number of overheard rebroadcasts and decreasing distance to neighboring rebroadcasting nodes. However, it takes time to collect these statistics, delaying the rebroadcast decision, hence a potentially high latency is introduced to every flooded message. In [25], Tseng *et al.* extend earlier approaches in [18] to allow nodes to dynamically adapt threshold values such as the rebroadcast counter.

In [28] Zhang and Agrawal propose an approach that is a combination of the counter-based and probabilistic method of [18]. Instead of using a static rebroadcast probability p, they adjust p according to the information collected by the counters. While this makes p adaptable, it becomes dependent upon other fixed parameters that need to be carefully selected (e.g., timeouts).

Dynamic Gossip [24] relies on local density awareness to adjust the rebroadcast probability p of the one-hop neighbors. Its correctness and suitability relies on the assumption that the nodes are uniformly distributed. Density information is collected using a relay-ping method.

In [15], Kowalski and Pelc propose a broadcasting algorithm with optimal lower bounds in their model. They consider only stationary nodes and ajust the broadcast probability accordingly.

Haas *et al.* [9] study what they term a *phase transition phenomena*. This work shows that purely probabilistic flooding (called *gossiping* in [9]) in an ad hoc network has a *bimodal* delivery distribution. Their simulations reveal that either almost every node receives the message, or virtually none. To reduce the likelihood of the latter case, they explore a variety of approaches, such as adapting the rebroadcast probability to the density or the distance to the flooding source. Sasson *et al.* [23] theoretically explore the same phenomena based on percolation theory and conclude that there exists a threshold $\bar{p} < 1$ such that for any $p > \bar{p}$ the node coverage is close to 1, while for $p < \bar{p}$ the coverage is very low. Hence, increasing p much beyond $\bar{p}$ is not very useful.

Any approach that bases rebroadcast decision on observation of neighbors and on overheard broadcasts is at risk of using stale information if nodes might move before the information is used. MANETs, of course, can have a high degree of mobility, hence neither of these approaches is ideal.

Mistral's compensation mechanisms is orthogonal to these approaches. Indeed, were we building a production deployment of flooding in a real-world setting, we would be inclined to combine Mistral with one of these others (as should be clear, the ideal choice of underlying mechanism depends upon the anticipated density of nodes and level of mobility; no single solution stands out as uniformly superior to the others). By using such a hybrid scheme, we could parameterize the underlying solution to keep overheads low, accepting a modest risk that flooded packets would fail to reach some nodes. Compensation packets could then be used to overcome this low level of residual losses.

3. MISTRAL

Traditional flooding suffers from the problem of redundant message reception, once per neighbor. Even in a reasonably connected network, the same message is received multiple times by every node, which is inefficient, wastes valuable resources, and can create contention in the transmission medium.

Selective rebroadcasting of flooded messages is a way to limit the number of redundant transmissions. Instead of simply rebroadcasting the message a node evaluates a local function $\mathcal{F}$ and then uses the outcome of this computation to decide whether to forward the message. In its simplest form, this function returns its result based on some static probability (corresponding to PPF). More complex functions take into account additional topological (e.g., the number of neighbors) or statistical information (e.g., the number of overheard rebroadcasts). The downside of selective flooding is that a flooding may no longer reach all intended nodes. In particular, if a node has only few neighbors, none of these neighbors may rebroadcast the message. Selective flooding thus balances message overhead against reliability.

Mistral finds some middle ground by introducing a new mechanism that allows us to fine-tune the balance between message overhead and reliability. The key idea is to extend selective flooding approaches by compensating for messages that are not rebroadcast. This compensation is based on a technique borrowed from forward error correction (FEC). Every incoming data packet (dp) is either rebroadcast or added to a compensation packet (cp). The compensation packet is broadcast at regular intervals and allows the receivers to recover one missing data packet.

3.1 Forward Error Correction

In its simplest form, Forward Error Correction (FEC) [11,

19, 22] creates l repair packets for every m data packets such that any m out of the resulting $(m+l)$ packets is enough to recover the original m data packets [11]. Traditional applications of FEC generate l repair packets for every m data packets and inject them into a data stream, which insulates the receiver from at most l packet losses. One of the fundamental advantages of FEC is that it imposes a constant overhead on the system and has easily understandable behavior under arbitrary network conditions. However, this simple form of FEC was developed for streaming settings, where a single sender is transmitting data at a high, steady rate such as in bulk file transfers [6] or in a video or audio feeds [7]. Part of our challenge is to develop a FEC solution matched to the characteristics of a MANET.

3.2 Algorithm

We noted earlier that Mistral can be built on top of any local-knowledge-based flooding approach. In the current implementation of the system, we use purely probabilistic flooding, mostly because this approach is extremely simple and is intuitively easy to visualize. Recall that in PPF, a node rebroadcasts a flooded message with static probability p. Although PPF might not be an ideal choice of algorithm in a practical deployment, the algorithm has no "hidden" effects that might make it hard to interpret our experimental findings.

Upon reception of a data packet, every node evaluates the function $\mathcal{F} : dp \longmapsto \{true|false\}$. In its most basic form, $\mathcal{F}$ takes a data packet as input and returns a boolean. If it returns true, dp is rebroadcast; otherwise, dp is added to the current compensation packet. When the number of data packets contained in a compensation packet passes a certain threshold c, the compensation packet is broadcast. We call c the *compensation rate*. Thus, a compensation packet is broadcast for every c data packets that are not rebroadcast.

Algorithm 1 presents the algorithm in more detail: Procedure `process` delivers the data packet to the application and decides whether to rebroadcast the packet or add it to the compensation packet; `composeCompensationPac` builds the compensation packet; and `runRecovery` attempts to recover data packets from stored compensation packets when a new data packet is delivered to the application. Finally, procedure `expand` is used for level-2 recovery, which is presented in Section 3.2.2. The secondary recovery mechanism discussed in the introduction is not included in Algorithm 1.

3.2.1 Composition of a Compensation Packet

In this section, we assume that data packets are of fixed size, e.g., 512 bytes, and contain the payload, a sender ID and some locally unique sequence number; we call these the *packet id*. The payload is assumed to remain unchanged during the course of the flooding (in some protocols, payloads do change as packets are routed; we discuss the handling of this kind of mutable payloads later in the paper).

To encode the payload of the data packets into the compensation packet, we use the XOR (operator $\otimes$), which is the simplest and best known FEC mechanism. A new data packet is added to the compensation packet by computing the XOR of its payload with the current payload in the compensation packet (initially, zero). Obviously, much more sophisticated error correction mechanisms are also possible; the advantage of XOR is its simplicity and low computational overhead.

Algorithm 1 Mistral's algorithm, code of node n_i.

```
1: Initialisation:
2:   DpBuffer ← ∅                                   {Received dps}
3:   cp ← ⊥                                         {Compensation packet}
4:   CpBuffer ← ∅                                   {Received cps}

5: upon flood(dp) do
6:   broadcast(dp)

7: upon reception of data packet dp for the first time do
8:   process(dp)
9:   runRecovery(dp)

10: upon reception of compensation packet cp from sender p_j
    do
11:   if cp.ids contains unknown dp ID then
12:     if recovery possible then
13:       dp_recov ← recover from cp
14:       process(dp_recov)
15:       runRecovery(dp_recov)
16:     else
17:       CpBuffer ← CpBuffer ∪ {cp}
18:       if level-2 recovery then
19:         expand(cp)
20:         for all recovered dp do
21:           process(dp)
22:           runRecovery(dp)

23: procedure process(dp)                {handles a data packet}
24:   DpBuffer ← DpBuffer ∪ {dp}
25:   if F(dp) then
26:     broadcast(dp)
27:   else
28:     composeCompensationPac(dp)
29:   deliver dp to the application

30: procedure composeCompensationPac(dp)    {constructs a
    cp}
31:   cp.payload ← cp.payload ⊗ dp.payload
32:   cp.ids ← cp.ids ∪ {dp.id}; cp.ttls ← cp.ttls ∪ {dp.ttl}
33:   if |cp.ids| ≥ c then{|X| returns the nbr of elements in
    X}
34:     broadcast(cp)
35:     cp ← ⊥

36: procedure runRecovery(dp)            {recovers dps from
    CpBuffer}
37:   for all cp1 ∈ CpBuffer do
38:     if dp.id ∈ cp1.ids then
39:       remove dp from cp              {including TTL and ID}
40:       if recovery from cp1 possible then
41:         dp_recov ⟵ recover from cp1
42:   for all recovered data packets dp'_recov do
43:     process(dp'_recov)
44:     runRecovery(dp'_recov)

45: procedure expand(cp)                 {level-2 recovery}
46:   for all cp1 ∈ CpBuffer do
47:     for all cp2 ∈ CpBuffer ∧ cp2 ≠ cp1 do
48:       if cp1 or cp2 is reducible then
49:         cp ← reduction from cp1 and cp2
50:         CpBuffer ← CpBuffer ∪ {cp}
```

If the receiver of a compensation packet already has all but one of the contained data packets, the compensation packet will allow the reconstruction of that missing data packet. However, the recipient of a compensation packet has no a-priori way to know what data packets were used to build the compensation packet. Accordingly, compensation packets must include a list of all its contained data packet IDs. Assuming IP-style node addresses, the sender ID is represented by four bytes. The local sequence number consists of one byte, which allows Mistral to send 255 flooded messages by a node before looping back to 0. From this, we can see that the size of a compensation packets will be the payload size plus five times the number of included data packets c, i.e, $|cp| = |payload_{dp}| + 5 * c$. Notice that the packet size is independent of the number of nodes in the system as a whole. This information is sufficient for floodings that span the entire node field.

A complication arises in applications where the the scope of flooding is limited by a time-to-live (TTL) parameter. Here, the compensation packets need to represent the TTL for each contained data packet; otherwise, if a node recovers data packet dp from a compensation packet, it has no way to know what TTL to use when rebroadcasting dp. If it chooses a TTL that is smaller than the true TTL, then the flooding may die out too early. If the TTL is too high, then valuable bandwidth is wasted. Even worse, if the flooding is a part of a routing mechanisms and the routing mechanism depends on the TTL, then loops occur in the routing paths.

Clearly we cannot treat the TTL of a data packet as a part of that packet's payload, since TTLs are decremented at every hop of the data packet. The problem here is that incoming TTLs for received packets might differ at the node undertaking the reconstruction relative to the node that built the compensation packet. Thus, TTLs need to be added to the compensation packet outside of the payload.

The simplest approach is to add a list of TTLs to the compensation packet. Since the TTL is generally represented by one byte another c bytes are added to the size of a compensation packet. In effect, the TTL extends the packet-id by one byte.

Unfortunatly, this simple approach adds additional overhead, which we would like to avoid. A first point to notice is that TTLs are often defined based on some estimate and are thus, by design, already an approximation. Hence, if we manage to limit the error to some low number, we can manage with an approximate recnstruction of the TTL value. For instance, we could store the sum of all TTLs. The TTL of a recovered data packet can then be restored by subtracting the TTL's of all known packets (all data packets except one). To limit the size to one byte, we apply the modulo operator to this sum. Using this approach, the error will in most cases be within ± 1, or in total $\pm c$, which is acceptable for most applications. Thus, the total size of a compensation packet is $5c + 1 + |payload_{dp}|$ bytes.

Although we have not explored the idea yet, it may be possible to further reduce the overhead associated with compensation packets by compressing packet-id information. For example, in a MANET where most communication originates with a very small set of senders, we could assign those senders some sort of very small id. Moreover, it may sometimes be possible to compress the compensation packet payload itself. On the other hand, such ideas increase the computational overhead at the receiver and hence would require careful evaluation.

3.2.2 Recovering from Compensation Packets

To recover data packets from compensation packets we use a two-level recovery mechanism. The first level recovers data packets based on the data packets that have already been received. If $c - 1$ data packets contained in a compensation packet are known, the missing one can be reconstructed. Compensation packets that contain two or more missing data packets are stored (in the $CpBuffer$) and reconsidered when new data packets arrive or are recovered from other compensation packets. Actually, we do not store complete compensation packets, but only compensation packets that contain the IDs, TTL(s), and payload of the missing packets. More specifically, we *xor* the known data packet payloads with the payload of the compensation packet. After some time compensation packets are garbage collected, as it has become highly unlikely that the missing data packet(s) will be received in the future.

The level-2 recovery mechanism is more elaborate. Instead of only considering incoming and recovered data packets this algorithm also matches compensation packets against each other. The matching operation works as a reduction. Each new compensation packet is compared with all stored compensation packets. If either one of the packets is completely contained in the other, then a new compensation packet is added, which contains the set of data packet IDs of the larger packet minus the ones in the smaller packet. The new payload is constructed by applying XOR to both compensation packets. Provided that it does not allow the immediate recovery of a data packet, this reduced compensation packet is then added to the set of stored compensation packets (in $CpBuffer$).

Clearly, level-2 recovery adds a considerable overhead, both in storage and computation. Its application thus makes sense only if the gain in recovered data packets is significant with respect to level-1 recovery. We explore level-1 and level-2 recovery using simulations in Section 5.2.3.

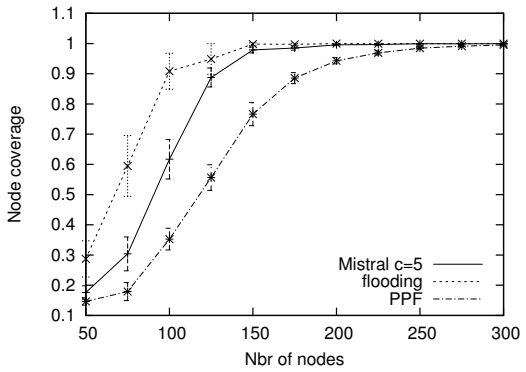
3.2.3 Mutable Payloads

Many routing protocols modify the flooded packets during the flooding. We have already shown how to handle TTL values. But some protocols modify other parts of data packets, for example by touching internal parameters, building a route trace, etc. To allow Mistral to handle these cases, we extend the above mechanism into compensation packets that include a mutable part and an immutable part of the payload. Clearly, the larger the immutable part is relative to the mutable part, the better the performance of Mistral. This is particularly the case as the immutable parts can be reduced into an immutable part of the same size, while mutable parts need to be appended to each other, thereby resulting in a size of $\sum_{i=0}^{c} mutablePartOf(dp_i)$. In genernal, the size of a compensation packet will now be $5c + |immutablePayload_{dp}| + \sum_{i=0}^{c} mutablePartOf(dp_i)$.

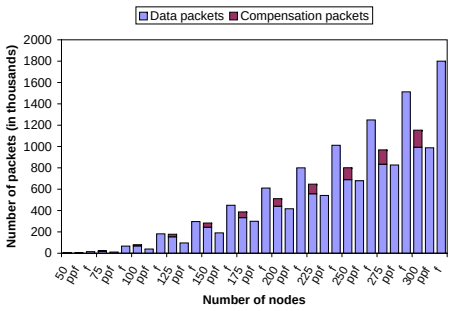
In the evaluation that follows, we assume that packets contain no mutable data other than the TTL.

4. ANALYSIS

In this section, we provide a simple analysis of Mistral. We denote by d_{max} the maximal diameter of the node field and consider floodings that span the entire node field. The maximal transmission latency $t_{maxTrans}$ is the maximal trans-



(a) Node coverage with varying density, $p = 0.55$.



(b) Message overhead with varying density, $p = 0.55$.

Figure 1: Node coverage and message overhead with varying node density.

mission range (88m) divided by the transmission speed. The time needed to do all the computations on a node is Δt, and we assume that there are no delays in the outgoing sending buffers, i.e., that there is no contention in the transmission medium.

Let f_i denote the number of floodings originating at node i, then the estimated overall generated number of compensation packets in a network with n nodes is $G = n\frac{(1-p)\Sigma_i f_i}{c}$, assuming that every node receives all flooded data packet at least once. Thus, the overhead in bytes is $G * (5c + 1 + |payload_{dp}|)$.

Assume that δ_{flood} denotes the average reception frequency of data packet that are received for the first time. Then, the estimated time needed to fill up a c-based compensation packet is $t_{recoveryPac} = \frac{c}{(1-p)*\delta_{flood}}$.

We now consider the delivery latency of a data packet. The worst case occurs when the flooding source and the destination are d_{max} hops apart and the data packet is always forwarded as part of a compensation packet. In this case, the maximum delivery latency is $d_{max} * (t_{recoveryPac} + \Delta t + t_{maxTrans})$, while the estimated maximum delivery latency is $(1-p) * d_{max} * (t_{recoveryPac} + \Delta t + t_{maxTrans}) + p * d_{max} * (\Delta t + t_{maxTrans})$.

We now compute the number of packets sent by a single flooding in a network of N connected nodes. Purely probabilistic flooding has a message overhead of $E(MsgOverhead) = p * N$, if we assume that every node receives the flooded message at least once. Mistral adds an estimate of $\frac{1}{c}$ for every dropped message. Thus, the total overhead per flooding is $(1-p) * N * \frac{1}{c} + p * N$. If the assumption that all nodes receive the flooded message is relaxed then the relative overhead added by Mistral increases. Each node that receives the flooded message only because of Mistral again contributes an additional broadcast or partial compensation to the overhead. Naturally, the additional overhead pays off through the increased node coverage.

5. SIMULATIONS

For our simulation we used JiST/SWANS v1.0.4 [1, 4], a simulation environment for ad hoc networks. Java applications written for a real deployment can be ported to the simulation environment and then placed under a variety of simulated scenarios and loads. JiST/SWANS intercepts the calls to the communication layer and dynamically transforms them into calls to the simulator's communication package.

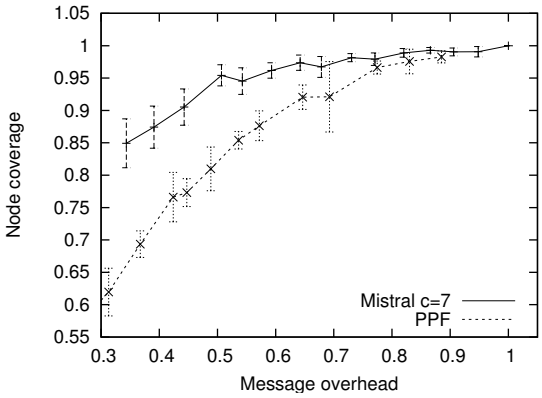
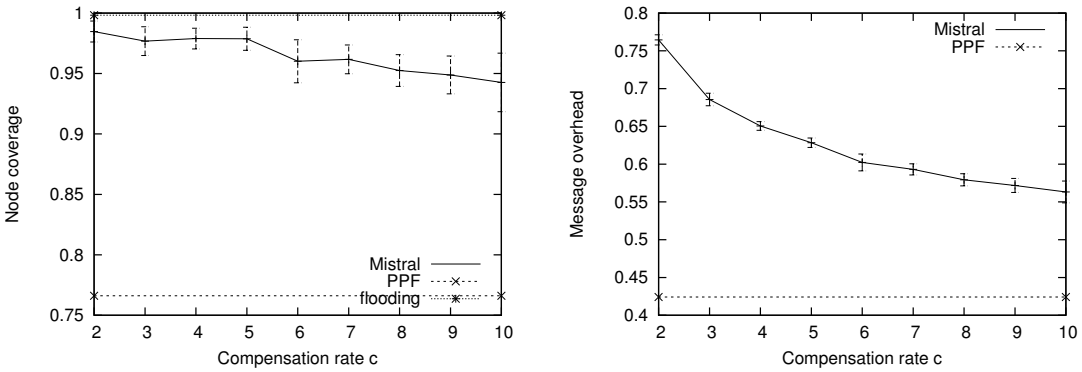


Figure 2: Node coverage with respect to message overhead.

5.1 Setup

We consider a set of nodes. Communication between two nodes m and n occurs in an ad hoc manner and may be asymmetric, i.e., n may be able to communicate with m, but the inverse may not be possible. Communication is by broadcast as defined in the 802.11b standard [12] and can be subject to interference, in which case the message cannot be received. Interference can occur without the sender being able to detect the problem (this is called the *hidden terminal problem* [2]).

We simulate a wireless ad-hoc network with 150 nodes uniformly distributed in a field of size 600x600m. Nodes are stationary, except for one case in which we measure the impact of mobility (Section 5.2.4). The maximal transmission range of a node is set to 88m. Every node starts flooding 20 messages at a regular interval, once all nodes are started up. All flooding occurs across the entire node field. Hence, ideally all nodes should receive all flooded messages.



(a) Node coverage. (b) Message overhead.

Figure 3: Varying compensation rate c, $p = 0.55$.

Our work models disconnections due to mobility, transmission range limits, and the hidden terminal problem just mentioned (using JiST/SWANS' *RadioNoiseIndep* package, which uses a radio model identical to ns2). Unless otherwise mentioned, we use the default values defined in JiST/SWANS.

The nodes start up at random times and positions. When they are all up and running, we start sending the flooding messages and we wait until all messages have been received (terminating simulation).

5.2 Results

In this section, we present the results of our simulation. Every node periodically, every 50s, floods a message throughout the entire field. We have chosen a low flooding rate because in our simulations we want to minimize the effect of packet loss due to buffer overflows and interference. The nodes are added to the sensor field at time 0s but start flooding at times uniformly distributed between 0 and 60s. All results give the average over at least 30 runs in different uniform node distributions. In general, the variance in the simulation results for ad hoc networks is high. This is due to the many sources of randomness: distribution of the sensor nodes, the paths of nodes, the time the nodes flood a message, etc. Thus, where significant we indicate the 95%-confidence intervals (CI).

To evaluate the quality of Mistral, we are mainly interested in two properties: node coverage and message overhead. *Node coverage* measures the number of nodes that have received the messages, while message overhead indicates the total number of sent messages. Both measurements are normalized against a connected network with the same number of nodes. In a connected network, any node can communicate with any other node. Thus, node coverage is given as a percentage of all nodes in the network, while message overhead is given as the percentage of the message overhead in the case in which all nodes receive all messages (normal flooding). Note that the message overhead in the connected network equals the product of the number of flooded messages with the number of nodes. Moreover, it is generally lower in a network with partitions. Since Mistral complements local-knowledge-based approaches and is based on purely probabilistic flooding, we compare Mistral to the latter. Purely probabilistic flooding is entirely defined by the rebroadcasting probability p. For completeness, we also show the results for simple flooding, which corresponds to PPF with $p = 1.0$.

In the following, we evaluate the following properties of Mistral: its behavior in the face of varying density, varying protocol parameters, node mobility, packet loss, and with the secondary recovery mechanism. Unless explicitly stated otherwise, we use the above default values in our measurements.

5.2.1 Impact of Density

We start by measuring the impact of node density on the node coverage and the message overhead. Fig. 1(a) shows the node coverage with varying number of nodes. It shows three measurements: simple flooding, purely probabilistic flooding (PPF), and Mistral with compensation rate $c = 5$. The rebroadcast probability is set to $p = 0.55$ in the cases of purely probabilistic flooding and Mistral. As expected, Mistral has a much higher node coverage than purely probabilistic flooding, especially for lower node densities. If the node density passes a certain threshold (around 225 nodes for Mistral), it is sufficiently high such that all nodes receive all messages. In contrast, with low density only a low percentage of the nodes receive all messages. However, below a certain threshold (around 150 nodes) even simple flooding cannot reach all nodes.

In Fig. 1(b) we show the corresponding message overhead. For every number of nodes indicated on the x-axis, we draw the sent number of packets for Mistral, purely probabilistic flooding (ppf), and simple flooding (f). Mistral's packets are further separated into data packets and compensation packets. Since Mistral adds additional compensation packets, its total message overhead is higher than the one of purely probabilistic flooding. Notice also that for low densities the number of flooding packets is higher. Due to higher node coverage in Mistral, more nodes receive the message and thus more nodes also rebroadcast the message, which accounts for the higher number of flooding packets compared to PPF.

Thus, to measure Mistral's net gain in node coverage, as compared to purely probabilistic flooding, we need to consider both node coverage and message overhead graphs. Indeed, since Mistral's compensation mechanism adds an additional overhead, we cannot directly compare the two approaches with the same rebroadcast probability p. Rather,

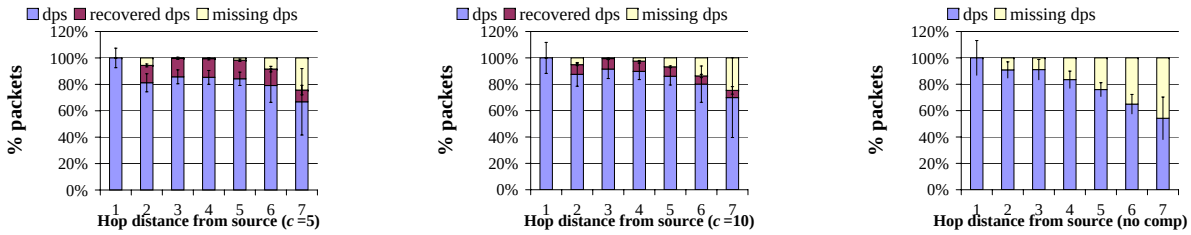


Figure 4: Recovery based on hop counts, single source and $p = 0.55$.

we need to compare Mistral with the purely probabilistic flooding using a rebroadcasting probability with a similar message overhead. Fig. 2 plots the node coverage with respect to the message overhead, for $c = 7$. The message overhead corresponds to simulation runs with p varying from 0.3 (0.4 for PPF) to 1, in steps of 0.05. The gain with Mistral is especially prominent for low rebroadcast probabilities p. Of course, low rebroadcast probabilities lead to many dropped rebroadcasts and thus the node coverage becomes low. Using Mistral allows some of the nodes to recover messages they may have missed. For an overhead of 0.35, Mistral improves the node coverage by 20%, for an overhead of 0.55 by 10%, and for overhead around 0.75 it is closer to 3%.

5.2.2 Compensation Rate

We now turn to one of the parameters that determine the behavior of Mistral: compensation rate c. In Fig. 3(a), we show the node coverage with compensation rate c varying from 2 to 10, for 150 nodes and rebroadcasting probability $p = 0.55$. Generally, the node coverage decreases with increasing compensation rate. For comparison, the graph also indicates the node coverage for flooding and purely probabilitstic flooding (PPF) with the same parameters. Both flooding and probabilistic flooding are independent from the compensation rate and thus are represented by a horizontal line. Fig. 3(b) gives the corresponding message overhead. Here, the message overhead decreases with increasing compensation rate. Thus, given a particular node coverage the higher the compensation rate the better. However, a higher compensation rate also increases the message delivery latency. Indeed, data packets that are part of a compensation packet spend more time waiting until the compensation packet is filled with sufficient data packets and may thus be delayed.

5.2.3 Recovery Performance and Overhead

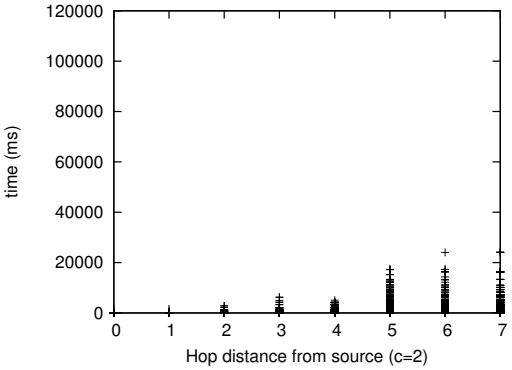
Next, we measure the number of recovered data packets with respect to the hop count (see Fig.4). In this simulation, a single node at position $[300, 300]$ periodically floods 1000 messages. We give the results for $c = 5$, $c = 10$, and the case with no compensation (no comp). Since the overall number of received data packets is different depending on the hop-distance of a node to the flooding source, we give the percentage of recovered data packets to all flooding packets that should have been received by the nodes at this hop distance from the source. The percentage of recovered data packets is approximately the same for most hop distances. An exception is at hop count 1, where all nodes generally receive the flooded message, because the source floods the data packet with $p = 1.0$ and at a time of low traffic. As fewer compensation packets are sent in the case of $c = 10$, the percentage of recovered data packets is lower compared to the case of $c = 5$. Towards very high numbers of hop counts, no compensation packets are received. However, these nodes are particular cases resulting from a unusual node distribution, which does not occur frequently.

Notice that the percentage of dps increases between $c = 5$ and $c = 10$. The reason is that with smaller c, more compensation packets are sent and the likelihood that a dps is received via a compensation packet increases. Since we count the data packets when they are received for the first time, more packets are received via a compensation packet. Thus, the percentage of dps is smaller for a smaller c.

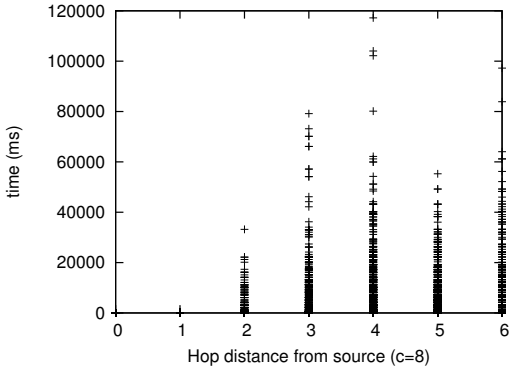
We use the same setup to measure the packet delivery latency. In contrast to the other simulations, we use a single data point (one random uniform distribution) in this case. The single source floods a data packet every second. The graphs in Fig. 5(a) and (b) show the latency distribution of data packets for $c = 2$ and $c = 8$ with respect to the hop distance of the node. The delivery latency of a data packet is high if it is received by a node only as part of a compensation packet. The higher the compensation rate, the higher this delay is.

Another important characteristic of Mistral is the ratio of compensation packets that cannot be recovered. We say that a *compensation packet is recovered* if all contained data packets have been received or have been recovered. In general, we expect the number of unrecovered compensation packets to increase with increasing compensation rate. The graph in Fig. 5(c) confirms this. It uses our default setup with many flooding sources and shows the total number of received compensation packets and the number of compensation packets that have not been recovered (logarithmic scale on y-axis). Clearly, the lower the compensation rate, the higher the number of sent and thus received compensation packets. This number also includes all compensation packets whose contained data packets have already been received earlier by the receiving node (useless cps). Immediately recovered compensation packets denote the compensation packets that only contain a single unknown data packet. Any compensation packet that contains more unknown data packets is added to *CpBuffer*.

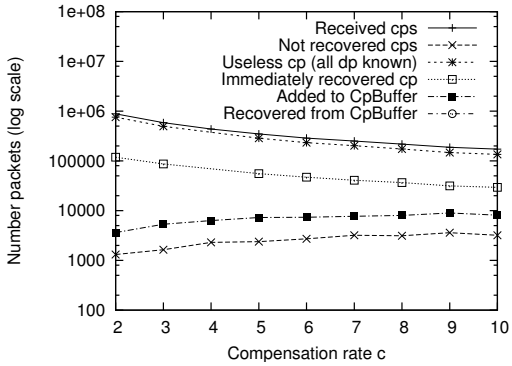
Fig. 5(d) shows the number of sent compensation packets based on the number of nodes in the field. As expected, this



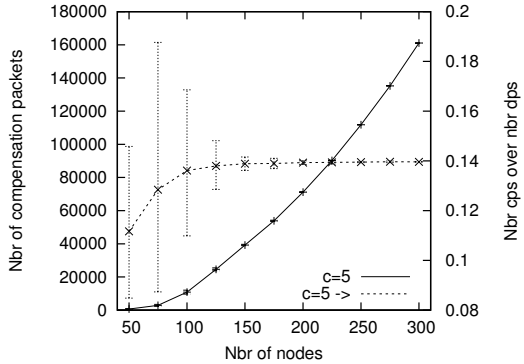
(a) Data packet delivery latency, single source.



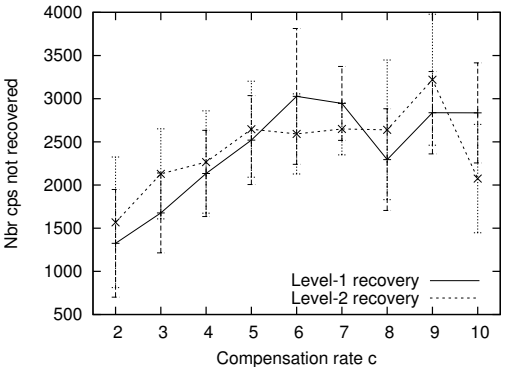
(b) Data packet delivery latency, single source.



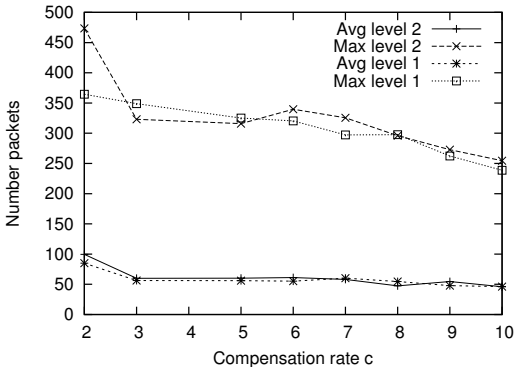
(c) Recovered compensation packets.



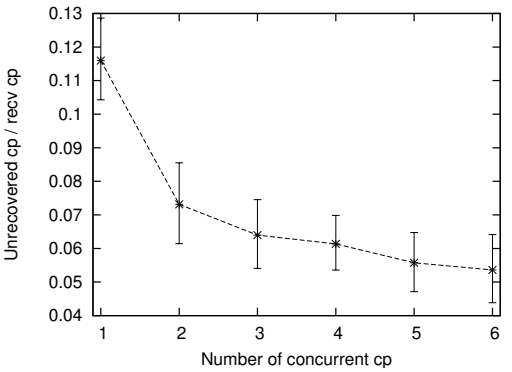
(d) Number of compensation packets with respect to node density, with $p = 0.55$.



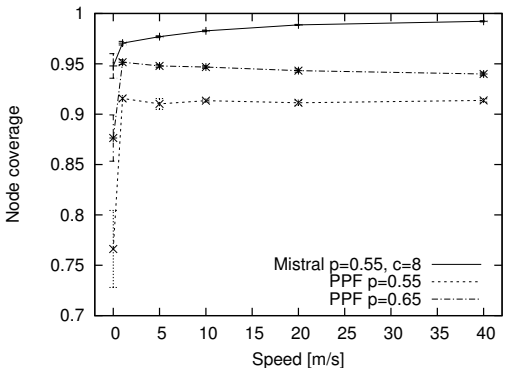
(e) Impact of stage 2 recovery, with $p = 0.55$.



(f) Memory requirement for stage 2 recovery.



(g)Varying number of concurrent cps, $p = 0.55$ and $c = 5$.



(h) Impact of node speed, with $p = 0.55$.

Figure 5: Recovery and node mobility.

curve is not linear. Rather, the more dense, and thus the more connected, the network is the higher the increase in the number of compensation packets becomes with respect to the next lower density. In comparison, the total number of broadcast data packets in the case of 300 nodes is 990000. In the same graph, we also show which percentage of all broadcast messages are compensation packets. For low densities, the percentage is relatively lower, as the nodes receive very few messages and may not be able to send the last compensation packet.

In Section 3.2.2, we described our two-level recovery algorithm: level 1 corresponds to the recovery mechanism based on incoming or recovered data packets, while level 2 extends the *CpBuffer* (see Procedure *extend* in Algorithm 1). We now measure the impact of level 2 recovery. Fig. 5(e) and (f) show the impact of level-2 recovery on node coverage and also regarding memory requirements. Our results show that level-2 recovery does not significantly increase node coverage. At the same time, it also has a similar memory overhead, in terms of the maximum and average number of compensation packets in *CpBuffer*. Thus, we conclude that in our setting level-1 recovery is sufficient for most applications.

Compensation packets work best if a node receives many different compensation packets that contain only a single unknown data packet, but a different one in every compensation packet. To get an indication of the impact of the distribution of data packets onto compensation packets, we artificially change the distribution in compensation packets. Rather than sequentially building one compensation packet after the other, we build a number of compensation packets in parallel and distribute the data packets randomly among them. We then measure the ratio of unrecovered compensation packets over the number of received compensation packets, with every node sending 50 flooded messages (Fig. 5(g)). This gives us an indication of the recovery rate as a function of the number of concurrently constructed compensation packets. Compared to the case of sequentially constructing a single compensation packet, Mistral achieves a considerable gain with even a moderate level of packet compensation concurrency.

5.2.4 Mobility

We now measure the impact of node mobility on node coverage. We use the random waypoint model [13] with a fixed speed and zero pause time, thereby removing the randomness caused by varying speeds and pause times.[1] In this model, nodes select an arbitrary location in the field and move there on a direct line (at constant speed in our case). When they reach the destination location, they wait for an arbitrary pause time (0s in our case) and then pick a new destination location. In the simulation in Fig. 5(h), we vary the node speed and show the impact on the node coverage. In the case of Mistral, node coverage increases with mobility. The same is the case with purely probabilitis flooding, in which the relative increase compared to the stationary case is much higher. However, when the speed increases with Mistral, node coverage also increases, which initially struck us as counter-intuitive. The explanation for this phenomenon turns out to be that Mistral adds a delay before sending compensation packets, especially with a large c, and thus packets tend to get distributed more in the node field.

Notice that we compare Mistral to PPF with a higher p. The rebroadcast probability p is selected such that the message overhead of PPF for stationary nodes corresponds to the one of Mistral. Since an increasing node coverage also increases message overhead and since PPF's relative node coverage increase between stationary and mobile nodes is much higher, PPF's message overhead also becomes higher for mobile nodes. Thus, the measured node coverage for PPF is an upper bound. In reality, the node coverage of a PPF with comparable message overhead would have $p < 0.65$ and thus also a lower node coverage.

The better node coverage with mobility stems from the fact that node mobility increases the likelyhood that a node overhears a broadcast at least at one of the positions it moves to. Furthermore, nodes do not stay in unfortunate distributions during a long time. The relative increase induced by node mobility is lower in the case of Mistral, because the node coverage is already much better for the stationary case.

Clearly, the increased node coverage is also related to the limits of the field. This has the consequence that the nodes move to much more positions in the field, thus the increased likelihood to receive the message. With unlimited field size, the nodes would just move outside the flooding area and thus no longer receive the flooded message.

5.2.5 Packet Loss

In Fig. 6, we show the impact of packet loss on node coverage. We assume that packets loss is uniformly distributed over all the packets in the system and occurs with the same probability at the sending and receiving side. Thus we remove 2k% packets, at the emission with k% probability and at the reception with k%. On the x-axis we indicate k. The graph shows that the node coverage decreases linearly, with a slightly higher decrease for PPF with comparable message overhead.

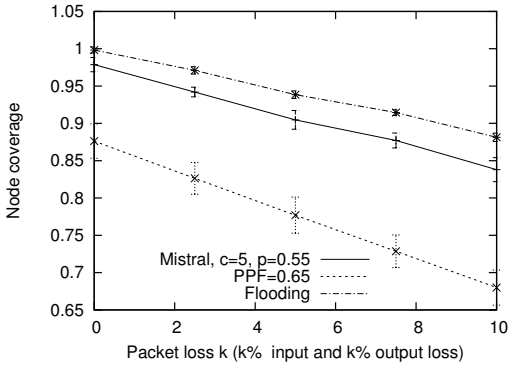


Figure 6: Varying packet loss rate.

5.2.6 Secondary Recovery Mechanism

We have mentioned that one advantage of compensation packets is that nodes learn what packets they might have missed. This information can be used to explicitly request missing packets from the neighbors. We have implemented this secondary recovery mechanism and present its impact on node coverage and message overhead in Fig. 7. For a

[1]Note that in [27] it has been shown that the random waypoint model is not entirely appropriate. However, for our measurements, this has no immediate impact.

moderate increase in message overhead, we gain significantly in node coverage.

6. CONCLUSION

The paper has presented an novel approach to flooding, based on the idea of proactively compensating for flooding packets that are not rebroadcast. Every compensation packet contains a tunable number of data packets. The receiver can use such a packet to recover a single lost data packet, provided that it has copies of the others used when constructing the compensation packet. The construction of compensation packets and the recovery of data packets rely on forward error correction mechanisms.

Mistral allows the application to tune the rebroadcast probabilities at a finer degree than purely probabilistic flooding. Compensation packets are sent only if the data packet is not rebroadcast. In addition to the rebroadcast probability p, the compensation mechanism can be fine-tuned using the compensation-rate parameter. Clearly, there is a tradeoff between delivery latency and the ability to recover a packets on one side, and message overhead on the other.

Mistral's compensation mechanism can support flooding by a wide range of concurrently active applications, and in this case the additional latency introduced by Mistral can be sharply reduced, because flooding by one application can also assist in the recovery of data lost in some other application. The only requirement is that the data packet payloads be of a similar size and that it be possible to pad any short packets.

We have implemented Mistral and then ported the runnable code to the JiST/SWANS platform, which allows us to take real code and then evaluate it in a simulated setting. Our simulation results show the improved node coverage of Mistral compared to purely probabilistic flooding with a similar overhead.

While we have investigated Mistral's compensation mechanism in the context of flooding scenarios, the same compensation mechanism could also applied to other applications where packet loss is an issue.

Although this paper limited itself to simulation, we do want to emphasize that Mistral is a real system and that the code we evaluated here is executable on real platforms with only minor modifications. Our hope in future work is to explore deployments of the system using a network of actual nodes.

7. ACKNOWLEDGMENTS

The authors are very grateful to Rimon Barr for his help on JiST/SWANS, Tudor Marian and Amar Phanishayee for their support with the cluster, and David Cavin for his comments on an earlier version of the paper.

8. REFERENCES

[1] JiST/SWANS. http://jist.ece.cs.cornell.edu.

[2] D. Allen. Hidden terminal problems in wireless LAN's. In *IEEE 802.11 Working Group Papers*, 1993.

[3] K. Alzoubi, P.-J. Wan, and O. Frieder. Message-optimal connected dominating sets in mobile ad hoc networks. In *Proc. of the 3th ACM Int. Symposium on Mobile Ad Hoc Networking & Computing (MobiHoc'02)*, pages 157–164, New York, NY, USA, 2002. ACM Press.

[4] R. Barr. *An efficient, unifying approach to simulation using virtual machines*. PhD thesis, Cornell University, Ithaca, NY, 14853, May 2004.

[5] V. Bharghavan and B. Das. Routing in ad hoc networks using minimum connected dominating sets. In *Proc. of the Int. Conference on Communications*, Montreal, Canada, June 1997.

[6] J. Byers, M. Luby, and M. Mitzenmacher. A digital fountain approach to asynchronous reliable multicast. *IEEE Journal on Selected Areas in Communications*, 20(8), Oct. 2002.

[7] G. Carle and E. Biersack. Survey of error recovery techniques for ip-based audio-visual multicast applications. *IEEE Network*, Dec. 1997.

[8] R. Gandhi, S. Parthasarathy, and A. Mishra. Minimizing broadcast latency and redundancy in ad hoc networks. In *Proc. of the 4th ACM Int. Symposium on Mobile Ad Hoc Networking & Computing (MobiHoc'03)*, pages 222–232, Annapolis, MD, 2003. ACM Press.

[9] Z. Haas, J. Halpern, and L. Li. Gossip-based ad hoc routing. In *Proc. of InfoCom 2002*, volume 21, pages 1707–1716, June 2002.

[10] C. Ho, K. Obraczka, G. Tsudik, and K. Viswanath. Flooding for reliable multicast in multi-hop ad hoc networks. In *3rd Intl. workshop on Discrete Algorithms and Methods for Mobile Computing and Communications (DIAL-M'99)*, pages 64–71, 1999.

[11] C. Huitema. The case for packet level FEC. In *Proc. of the TC6 WG6.1/6.4 5th Int. Workshop on Protocols for High-Speed Networks (PfHSN'96)*, pages 109–120, London, UK, 1996. Chapman & Hall, Ltd.

[12] IEEE. 802.11 specification (part 11): Wireless LAN medium access control (MAC) and physical layer (PHY) specifications, June 1997.

[13] D. Johnson and D. Maltz. Dynamic source routing in ad hoc wireless networks. In Imielinski and Korth, editors, *Mobile Computing*, volume 353. Kluwer Academic Publishers, 1996.

[14] Y.-B. Ko and N. Vaidya. Flooding-based geocasting protocols for mobile ad hoc networks. *Mobile Networks and Applications*, 7(6):471–480, 2002.

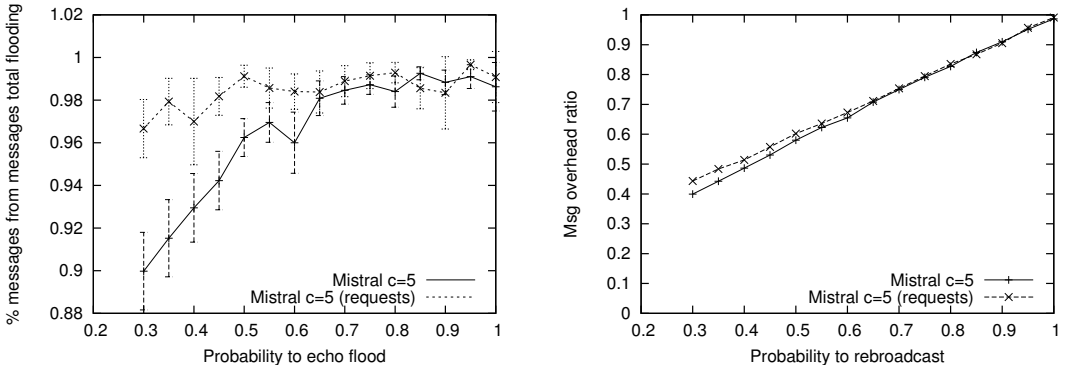
[15] D. Kowalski and A. Pelc. Broadcasting in undirected ad hoc radio networks. pages 73–82, New York, NY, USA, 2003. ACM Press.

[16] S. Lee, W. Su, and M. Gerla. On-demand multicast routing protocol in multihop wireless mobile networks. *Mobile Network Applications*, 7(6):441–453, 2002.

[17] W. Lou and J. Wu. Double-covered broadcast (DCB): A simple reliable broadcast algorithm in MANETs. In *Proc. of INFOCOMM*, 2004.

[18] S.-Y. Ni, Y.-C. Tseng, Y.-S. Chen, and J.-P. Sheu. The broadcast storm problem in a mobile ad hoc network. In *Proc. of the 5th ACM/IEEE Int. Conference on Mobile Computing and Networking (MobiCom '99)*, pages 151–162, New York, NY, USA, 1999. ACM Press.

[19] J. Nonnenmacher, E. Biersack, and D. Towsley. Parity-based loss recovery for reliable multicast transmission. *IEEE/ACM Transactions on Networking*, 6(4):349–361, 1998.



(a) Node coverage. (b) Message overhead.

Figure 7: Explicitly requesting missing packets.

[20] W. Peng and X.-C. Lu. On the reduction of broadcast redundancy in mobile ad hoc networks. In *Proc. of the 1st ACM Int. Symposium on Mobile Ad Hoc Networking & Computing (MobiHoc'00)*, pages 129–130, Piscataway, NJ, USA, 2000. ACM.

[21] C. Perkins and E. Royer. Ad-Hoc On Demand Distance Vector Routing. In *Proceedings of the IEEE Workshop on Mobile Computing Systems and Applications (WMCSA)*, Feb. 1999.

[22] L. Rizzo and L. Vicisano. A reliable multicast data distribution protocol based on software FEC techniques. In *Proc. of the 4th IEEE Workshop on the Architecture and Implementation of High Performance Communication Systems (HPCS'97)*, Sani Beach, Chalkidiki, Greece, June 1997.

[23] Y. Sasson, D. Cavin, and A. Schiper. Probabilistic broadcast for flooding in wireless mobile ad hoc networks. In *Proceedings of IEEE Wireless Communications and Networking Conference (WCNC 2003)*, Mar. 2003.

[24] D. Scott and A. Yasinsac. Dynamic probabilistic retransmission in ad hoc networks. In *Proc of the Int. Conference on Wireless Networks (ICWN'04)*, pages 158–164, Las Vegas, Nevada, June 2004. CSREA Press.

[25] Y.-C. Tseng, S.-Y. Ni, and E.-Y. Shih. Adaptive approaches to relieving broadcast storms in a wireless multihop mobile ad hoc network. In *Proc. of the 21st Int. Conference on Distributed Computing Systems (ICDCS'01)*, pages 481–488, Phoenix, Arizona, Apr. 2001.

[26] B. Williams and T. Camp. Comparison of broadcasting techniques for mobile ad hoc networks. In *Proc. of the ACM Int. Symposium on Mobile Ad Hoc Networking and Computing (MOBIHOC'02)*, pages 194–205, 2002.

[27] J. Yoon, M. Liu, and B. Noble. Random waypoint considered harmful. In *INFOCOM 2003*, Apr. 2003.

[28] Q. Zhang and D. P. Agrawal. Dynamic probabilistic broadcasting in manets. *J. Parallel Distrib. Comput.*, 65(2):220–233, Feb. 2005.