Information Dissemination in Highly Dynamic Graphs *

Regina O'Dell
odell@tik.ee.ethz.ch

Roger Wattenhofer
wattenhofer@tik.ee.ethz.ch

Computer Engineering and Networks Lab
ETH Zurich, Switzerland

ABSTRACT

We investigate to what extent flooding and routing is possible if the graph is allowed to change unpredictably at each time step. We study what minimal requirements are necessary so that a node may correctly flood or route a message in a network whose links may change arbitrarily at any given point, subject to the condition that the underlying graph is connected. We look at algorithmic constraints such as limited storage, no knowledge of an upper bound on the number of nodes, and no usage of identifiers. We look at flooding as well as routing to some existing specified destination and give algorithms.

Categories and Subject Descriptors

F.2.2 [**Analysis of Algorithms and Problem Complexity**]: Nonnumerical Algorithms and Problems

General Terms

Algorithms, Theory

Keywords

flooding, routing, mobile network, dynamic graphs

1. INTRODUCTION

1.1 Motivation

For mobile ad hoc networks, there are possibly as many routing protocols as there are mobility models. Virtually every newly introduced mobile routing algorithm is first described in terms of the ideas leading to its design and then tested in simulation for some form of assumed mobility. Or the mobility under consideration is *coarsely grained*: When routing is in progress, the network is relatively stable to allow for the search and usage of a path between the source and the destination. Unfortunately, this process will lead only slowly to insights into the fundamental principles underlying mobile computing. It is clear that dynamic networks pose challenges both to the engineering as well as algorithmic community. We are interested to what extent these challenges are inherent to the mobility and ad hoc nature of a network and when they can be overcome. In particular, we are interested in the details of how high mobility and algorithmic constraints affect what can be computed in an ad hoc network.

Traditional distributed algorithms on static graphs focus on the issue of *locality* which refers to the direct neighborhood of a node. In that context, Naor and Stockmeyer [10] ask: What can be computed locally? Put another way, given a graph theoretic problem, how much can one limit the resources and/or the number of communication rounds and still be able to compute it? If a network is highly mobile, then the view that a node has is also restricted to its local, direct neighborhood because the graph might change too quickly to gather any more information. In that case, we have two other obstructions: (i) from the inside, we limit the knowledge an algorithm can have a priori and (ii) from the outside, the challenge posed by the mobility of the network. Then we ask: What can be computed at all? That is, in a first step, we are not so much interested in the time and message complexity of the task, but its solvability.

We consider graphs that are highly mobile. Already Heraclitus of the ancient Greeks proclaimed the philosophy of *panta rhei*, everything flows. In that spirit, we allow the graph to change at every single step, that is, after each message transmission. We make no assumptions about how long it takes for a message to transfer or how long a path remains stable, only that the graph locally cannot change faster than it takes for a single message to be delivered. We could term this *fine-grained mobility*. We only impose that the graph be connected at all times so that an algorithm has a reasonable chance of success. Put another way, we let the mobility of the nodes be fast enough such that there is no guarantee for a node to send a query to its neighbor and wait for an answer, and slow enough such that when a new neighbor arrives, it will be able to receive a message from its new neighborhood.

A node must thus be able to detect when its neighborhood changes. In practice, one can imagine that a moving node will change the ambient temperature of the other nodes, or it

*The work presented in this paper was supported (in part) by the National Competence Center in Research on Mobile Information and Communication Systems (NCCR-MICS), a center supported by the Swiss National Science Foundation under grant number 5005-67322.

DIALM-POMC'05, September 2, 2005, Cologne, Germany.

will leave a visible dust trail, or any number of other physical indications. Alternatively, one can design a node which is able to detect its own motion and can notify its neighbors by a special beacon that it is moving and the neighborhood might change. There are numerous ways one can conceive of which avoid the periodic beaconing of keep-alive messages.

On the other hand, we cannot expect algorithms designed for large scale ad hoc and sensor networks to be omnipotent. The very nature of these networks implies that the nodes only have limited knowledge about the network, such as its size. If one considers a network deployed over an extended period of time, such as in habitat monitoring, then the number of nodes can fluctuate greatly over the years as old ones die and new ones are installed. Some sensor networks need to employ tiny nodes with limited capabilities, in which case the storage overhead of the algorithms becomes another critical issue.

In this paper, we focus on flooding and routing in highly mobile ad hoc wireless networks. Routing is a basic problem of any network where information needs to be transported from one place to another. In sensor networks, this is typically the reading from some sensor node which needs to be relayed to a sink who then catalogues the data. As the network changes, any routing information can become stale, and flooding is a fundamental ingredient of any routing algorithm to find a valid path to the destination. Flooding (or broadcast) is also of independent interest when a node wishes to distribute some vital information to the entire network. Clearly, this can be achieved if every node simply broadcasts the message, once received, to its neighbors at all times. However, if we consider the limited resources at the nodes' disposal, this means a tremendous energy consumption. Therefore, *termination* – in the very least – of any algorithm must be guaranteed.

1.2 Related Work

When it comes to routing in mobile ad hoc networks, a vast multitude of protocols have been proposed in the literature, see [9] for an overview. These algorithms may base their decisions only on the available topology of the graph or also on the (relative) coordinates of the nodes. For these latter geometric routing algorithms, it is known that greedy (i.e., local) forwarding strategies may lead into dead-ends, while optimal guaranteed delivery has only been shown for the static case [8] because it involves a preprocessing stage. Some steps in the direction of mobile (geometric) routing are taken in [6]. If we consider topology-based routing algorithms, only reactive protocols such as DSR [7] or AODV [12] make sense in a highly mobile environment. FRESH [2] also takes mobility into account, but not in the worst-case sense. Its refreshing perspective is that it sees mobility as a resource rather than a handicap. There also exist hybrid protocols, most notably ZRP [5] and its successor IZR [13]. Yet, to the best of our knowledge, none of these protocols take into account mobility at every single moment and from a worst-case perspective. For link-reversal type algorithms, it has been shown already in [4] that the routing will stabilize *eventually* if the graph itself remains stable for a finite amount of time. If the network continually changes, then a packet might roam around the network indefinitely.

In a highly dynamic scenario, the only thing that nodes can know with certainty about the graph is their direct neighborhood at each time step. Thus, highly mobile graph algorithms are related to local graph algorithms as discussed above. For instance, Peleg [11] discusses the complexity of broadcast in a network with limited message sizes and limited knowledge of the nodes. If nodes do not know their neighborhoods, then any broadcast algorithm needs messages in the order of the number of edges in the graph (counting one message per edge). This is also the case where nodes do know their direct neighborhood, but message size is restricted to $O(\log n)$. Many such other lower bounds are given in [3]. However, this applies only to static graphs. In dynamic graphs, the first question that begs itself is one of feasibility; complexity is of secondary importance.

1.3 Outline

We will first introduce and discuss our network model formally in Section 2.1. Next, we specify in Section 2.2 what the capabilities of an ad hoc mobile algorithm should be. We list four basic requirements along with their motivations which a scalable and adaptable algorithm must fulfill. In order to gain an understanding of the problem, we will first give algorithms that meet three out of the four conditions in Section 3. Generally, the techniques developed there will pave the way for the remaining flooding and routing algorithms. Then, in Section 4, we will see what happens to the flooding problem if we require the algorithm to satisfy all four restrictions.

Finally, in Section 5, we give a routing algorithm for the case when flooding seems to be impossible which is when the usage of identifiers is not allowed. One might counter that the routing problem needs a *destination identifier*. But, we can think of cases where in a sensor network, there is only one destination which is designated by a unique identifier, such as 0, that no other node will take on. Or, we might consider routing to a node with a specific service, such as relaying the message to a node with Internet access.

2. MODEL AND NOTATION

2.1 Network and Mobility Model

The graph $G = (V, E)$ is allowed to be mobile in the following sense. At all times t, $V_t = V(G)$ remains the same but $E_t \subseteq \binom{V}{2}$ is arbitrary subject to the condition that G_t is connected. If the connectivity of G is not guaranteed, then a node might have to hold a message indefinitely if we do not know when G will become connected again. Formally, we can consider the dynamic graph G as a sequence of edge changes $E = E_{t_0}, E_{t_1}, E_{t_2}, \ldots$

Message transmissions are asynchronous and we assume that local processing time is negligible. All messages are sent as broadcasts to the neighborhood as motivated by the shared medium of wireless networks. The speed at which messages travel limits the frequency of graph changes. Let $N_{t_0^v}(v), N_{t_1^v}(v), \ldots$ be the sequence of neighborhoods for node v with $N_{t_{i+1}^v}(v) \neq N_{t_i^v}(v)$ for all $i \geq 0$ and $N_{\tilde{t}}(v) = N_{t_i^v}$ when $t_i^v \leq \tilde{t} < t_{i+1}^v$. Assume that the maximum time it takes to transmit a message is T. Then we require that

$$T \leq \min_i \{t_{i+1}^v - t_i^v\} \qquad \forall v \in V \tag{1}$$

that is, the graph locally cannot change faster than it takes for a message to transmit. The idea of this model of mobility is that we need the following: If a node w enters the neighborhood of node v upon which v immediately sends a

message, then w is guaranteed to receive it. Note that the formalization in Eq. (1) is actually more restrictive than the informal description just given.

Other than the receipt of a message, nodes need to know about a local topology change as an event since we do not insist on an algorithm to operate in synchronous rounds. We assume that every node v is able to detect a change in its neighborhood $N(v)$, that is, when $\Delta N(v) \neq \emptyset$. Observe that we do not require the nodes to be able to learn their actual neighbors at all times, merely a local change in the graph.

Note that we allow the events $\Delta N(v) \neq \emptyset$ and "message receipt" to occur *simultaneously*. This implies that a node can receive a message, and then immediately its neighborhood changes before it broadcasts the message. Further note that this model also allows for messages to become "lost." This may happen if the graph around a node v has been stationary for an extended period of time (i.e., longer than T). When v transmits a message, nodes may move out of its neighborhood before the message has arrived at any of them, and new ones may have arrived without hearing the message as well. However, when the neighborhood changes trigger an event at v, all those nodes will receive the subsequent transmission.

LEMMA 1. *If a node v transmits a message at time t and also whenever $\Delta N(v) \neq \emptyset$, then at least one node will receive v's message before time $t + 2T$.*

To simplify notation, we will consider the neighborhood of a set of nodes as

$$N(X) = \bigcup_{x \in X} N(x) \quad \text{for } X \subseteq V.$$

Furthermore, we often consider a subset of the nodes with a certain property over time, such as $N_t(v)$ is the set $N(v)$ of neighbors of v at time t. If the time is irrelevant or clear from the context, then we drop the subscript.

2.2 Algorithmic Goals

For any routing or information-disseminating algorithm, we impose the following

REQUIREMENTS. Any dynamic graph algorithm should achieve

1. Correctness,
2. Termination.

Correctness depends on the task of the algorithm. If we look at routing, then the destination must be reached (assuming it exists). If we look at flooding, then all nodes must be reached. The *termination* condition means that eventually no node will transmit any more messages.

CONDITIONS. A dynamic graph algorithm may have to operate under the following conditions:

1. $O(\log n)$ bits storage and message overhead,
2. Uniform network.

We assume that nodes have *unique* identifiers with size polynomial in n, implying that they can be represented in $O(\log n)$ bits. In other words, Condition 1 about *storage space* allows a node to remember only a constant number of other nodes' IDs. We will examine what happens if we do not have such a manageable identifier space in Sections 4 and 5. The storage-space condition is another reason why we let an algorithm only detect whether the neighborhood changes and not store the entire neighborhood list which might be in the order of n entries.

The *uniformity* restriction is perhaps the strictest or at least the most questionable of the above. Specifically, an algorithm knows neither the exact number of nodes n nor an upper bound on n. The first two requirements are natural, the storage condition is reasonable, especially when dealing with sensor nodes on an increasingly smaller scale and greater coverage. Uniformity is more uncommon which is why we will first investigate how straightforward flooding is without it in Section 3. However, we argue that this restriction is not unreasonable or unrealistic. For example, once a sensor network to monitor wildlife is in place, perhaps after a while the population of the animals and with it the number of deployed sensors grows drastically beyond expectation, then any previous assumption on n will become invalid. Or, if sensor nodes will be deployed in space, then the room for growth there is inexhaustible for all intents and purposes. In other words, not knowing a bound on n allows for indefinite growth of the network.

As we are interested in gauging the realm of possibility for mobile algorithms, formulating these conditions separately will allow us to determine which requirements are essential and which are merely "decoration."

3. "THREE OUT OF FOUR" FLOODING

Observe that instead of simultaneously satisfying all of the above requirements and conditions, we can consider dropping each one separately to see if the problem is easily solvable in that case. If we drop either of the Requirements 1 or 2, then the task of flooding becomes trivial. If we do not want correctness, then simply do not send a message (or just send it once). If we do not want termination, then simply retransmit the message infinitely many times. Dropping either of the conditions will make us do a little more work and is the content of this section.

If we do not impose Condition 2, we have the following. COUNTERFLOODING is described in Algorithm 1.

Algorithm 1 COUNTERFLOODING.

Input: n
Events: receipt of message, $\Delta N \neq \emptyset$
1: `receive` msg
2: `broadcast` msg
3: $k \leftarrow 0$
4: **if** $(k < 2n) \wedge (\Delta N \neq \emptyset)$ **then**
5: `broadcast` msg
6: $k \leftarrow k + 1$
7: **end if**

LEMMA 2 (DROPPING COND. 2). *If a polynomial upper bound on the number of nodes $n \geq |V|$ is known, then the algorithm* COUNTERFLOODING *achieves Requirements 1 and 2 under Condition 1.*

PROOF. Denote by $I_t \subset V$ the set of idle nodes which have not seen the message on or before time $t \geq 0$. Since $|I_0| < n$ (setting $t = 0$ when the node s starts the flooding),

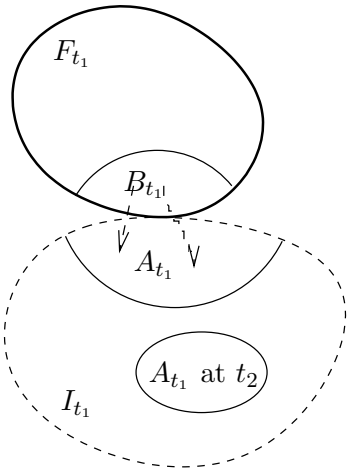

Figure 1: See the proof of Lemma 2.

we need to show that the total time to reach all nodes is less than $2Tn$, T as in Eq. (1), which implies that there are less than $2n$ neighborhood changes per node. Thus, $k_t(v) < 2n$ for all $v \in V$ while there are idle nodes.

We claim that

$$|I_{t+2T}| < |I_t| \neq 0 \tag{2}$$

for $I_t \neq \emptyset$ where T is the maximum time it takes to transmit a message. Let A_t be those idle nodes bordering the non-idle ones, specifically,

$$A_t = \{v \in I_t \mid N_t(v) \setminus I_t \neq \emptyset\}.$$

Analogously, let B_t be the non-idle nodes bordering the idle ones:

$$B_t = \{v \in F_t \mid N_t(v) \cap I_t \neq \emptyset\}$$

denoting by $F_t = V \setminus I_t$ the set of flooding, or non-idle, nodes. Observe that $B_t \neq \emptyset$ for all times t as G_t is connected.

Fix a time $t \geq 0$ and take nodes $v \in A_t$ and $w \in B_t$ such that $v \in N_t(w)$. Now consider the latest time $t_1 \leq t$ such that $v \in N_{t_1}(w)$ but $v \notin N_{\hat{t}}(w)$ for $\hat{t} < t_1$. Note that possibly $t_1 = 0$, meaning that v and w have been neighbors from time 0 until t. Let $t_2 \leq t$ be the time when w received the message for the first time.

If $t_1 > t_2$, then the arrival of v into w's neighborhood triggered a $\Delta N_{t_1}(w) \neq \emptyset$ event and w will forward the message to v by Lines 4 and 5, arriving at some time $t' \leq t_1 + T \leq t + T$, thus $v \in I_t$ but $v \notin I_{t+2T}$.

If $t_1 \leq t_2$, then w will broadcast the message by Line 2. The only way for v *not* to receive the message is if $v \notin N_{t_3}(w)$ at some time $t_2 \leq t_3 < t_2 + T$. By our definition of t_1, we have $v \in N_{t'}(w)$ for all $t_1 \leq t' \leq t$. This implies that $t_3 > t$ as $t_2 \geq t_1$, in which case $t < t_2 + T$. This can happen to all the nodes in A_t, such that $N_{t_3}(A_t) \setminus I_{t_3} = \emptyset$. (If it does not, then the remaining nodes in A_t will receive the message on or before time $t_2 + T \leq t + T$ and we are done.) Then also $I_t \setminus A_t \neq \emptyset$.

As G_{t_3} is connected, there is an idle node $u \in I_t \setminus A_t$ such that $u \in A_{t_3}$ with a neighbor $x \in B_{t_3}$. This means that u will trigger a $\Delta N_{t_3}(x) \neq \emptyset$ event, because u was not neighboring x at time t but subsequently at time $t_3 > t$. Then $u \notin I_{t+2T}$ as it will receive x's message on or before $t_3 + T < t_2 + 2T \leq t + 2T$. □

We then immediately have the following.

COROLLARY 3. *Algorithm* COUNTERFLOODING *achieves correctness after time at most $2Tn$ where T is the maximum message transmission time. In fact, the input parameter n guarantees that at least n nodes (if that many exist) will hear the message sent from a starting node s.*

COUNTERFLOODING is a key ingredient to the remainder of the paper. The idea is that if the nodes do not have exact knowledge of n, then a common trick of the subsequent algorithms is to estimate $|V|$ by some other means and then execute COUNTERFLOODING. The crucial point is that the nodes need to ensure that their estimate of $|V|$ will keep increasing, so that eventually it will be greater than $|V|$ or all the nodes already heard the message anyway.

Then what if we only leave out Condition 1 and allow for unlimited storage and message overhead? To that end, consider the following algorithm: Each node v maintains a list l_v of nodes which it knows to have heard the message. Initially, $l_v \leftarrow \{v\}$. Every time node v broadcasts the message, v includes its list l_v in the header. Every time v receives a message from some node w, it merges the lists $l_v \leftarrow l_v \cup l_w$ before proceeding. The length of the list defines the estimate of the number of nodes that there are, that is, $\hat{n}_v = |l_v|$. Node v restarts Algorithm 1 with $n = \hat{n}_v + 1$ each time l_v increases. The algorithm is started by s, thus $s \in l_v$ for all flooding v. Pseudo code for LISTFLOODING is given in Algorithm 2. Observe that when calling COUNTERFLOODING as a subroutine, we drop Line 1 and have a modified message as additional input.

Algorithm 2 LISTFLOODING at node v

Input: none
Events: receipt of message, $\Delta N \neq \emptyset$
1: $l \leftarrow \{v\}$, $\hat{n} \leftarrow |l|$
2: `receive` (msg,l_w)
3: $l \leftarrow l \cup l_w$
4: **if** $|l| > \hat{n}$ **then**
5: $\quad \hat{n} \leftarrow |l|$
6: $\quad$ COUNTERFLOODING $(\hat{n}+1)$, append l to msg
7: **end if**

LEMMA 4 (DROPPING COND. 1). *If we allow for storage and header size in $O(n \log n)$, then algorithm* LISTFLOODING *achieves Requirements 1 and 2 under Condition 2.*

PROOF. We will argue that the list of flooding nodes F will continue to increase after a finite amount of time. Correctness is then guaranteed because either the maximum list of reached nodes continues to grow until it contains all the flooding nodes and then finally spreads to the idle nodes or else an idle node is reached before that.

Now we will prove the claim that F will increase after a finite amount of time until it encompasses the entire set of nodes. Consider the time t when a list in G is increased to l_{max} at node v_{max}, the maximum sized list of the flooding nodes F_t. Initially, this happens at $t = 0$ when $s = v_{max}$ starts the flooding and $l_s = \{s\}$. With the arrival of the next message at $\tilde{t} \leq t + 2T$ (Lemma 1), three things can happen:

(i) $|F_{\tilde{t}}| > |F_t|$, in which case we are done;

(ii) $|l_{max,\tilde{t}}| > |l_{max,t}|$, then set $t = \tilde{t}$ and restart the argument; or

(iii) neither of the above.

For case (iii), observe that all receivers of v_{max}'s transmission will have an equal estimate $\hat{n} = |l_{max}|$. Thus, at time t, we are starting Algorithm 1 with $n = \hat{n}+1$. After time at most $2Tn$, by Cor. 3, we have reached $\hat{n}+1$ nodes and by the pigeonhole principle, there is at least one of those, call it u, such that $u \notin l_{max}$. Therefore, $l_{max,t'} \supseteq l_{max,t} \cup \{u\}$ for $t' \leq t + 2T(\hat{n}+1)$.

It may be that there are several nodes at time t with a maximum list. If their lists are the same, then the case above with $l_v = l_{max}$, $\forall v \in l_{max}$, will happen even sooner. If they are different, then we can consider their flooding separately and by the above argument the individual list clusters will still grow.

To ensure termination, observe that $l_{max} \subseteq V$ which means after time $O(Tn^2)$, l_{max} cannot grow anymore and therefore, no counters will be reset. □

Interestingly enough, dropping the space condition leaves us with a much slower algorithm than when we drop the uniformity condition.

COROLLARY 5. *Algorithm* LISTFLOODING *achieves correctness after time $O(Tn^2)$ where T is the maximum message transmission time.*

THEOREM 6. *We can devise an asynchronous flooding algorithm fulfilling any three out of Req. 1, Req. 2, Cond. 1, or Cond. 2.*

4. FLOODING

Now that we understand flooding in the context of highly dynamic networks a little better, we will ask about the existence of a flooding algorithm which fulfills *all* the requirements. First we look at it from the original model where nodes have small and unique IDs. We could alternatively imagine the use of a random ID generator with sufficiently small collision probability, in which case the algorithm given below will be correct with high probability. Next, we question that assumption and look at flooding without the use of IDs.

4.1 Using Identifiers

The trick from the flooding algorithm before was to keep a list of nodes which have received the message which in turn allows the algorithm to progressively learn about the number of nodes in G. If we have identifiers available from 1 to n^d for some constant d, then the highest ID will be an upper bound on n. In other words, we can try to find the maximum ID in the network, and then use COUNTERFLOODING to ensure that all nodes are reached. The only difficulty lies in the starting node s reaching the highest ID node. The reasoning from LISTFLOODING gives us the necessary insight to see that this is possible.

THEOREM 7. *Algorithm* IDFLOODING *fulfills all Requirements 1 and 2 under Conditions 1 and 2.*

PROOF. The reasoning is the same as in the proof of Lemma 4, except that now the maximum node ID gives us the estimate of n. We give the main inductive argument.

Algorithm 3 IDFLOODING at node v

Input: none
Events: receipt of message, $\Delta N \neq \emptyset$

```
1: n̂ ← 0
2: receive (msg,w)
3: w ← max(v, w) {needed for first message}
4: if w > n̂ then
5:    n̂ ← w
6:    COUNTERFLOODING (n̂ + 1), append n̂ to msg
7: end if
```

Consider the so-far highest node ID x of the flooding nodes. By Lemma 2, we will reach $x+1$ nodes after x starts. Therefore, at least one of those will have a higher ID than x and we make progress either on the highest circulating ID or on the number of informed nodes. Once the node with the highest ID receives the message, its call of COUNTERFLOODING will reach all the nodes. Termination is guaranteed by the existence of a node with the highest ID. Small storage is ensured by the polynomial-sized identifier space. □

COROLLARY 8. *Algorithm* IDFLOODING *achieves correctness after time $O(Tn^2)$ where T is the maximum message transmission time.*

4.2 Without Identifiers

Some might argue that the existence of such identifiers is questionable. For instance, consider a sensor network in some remote place over a long period of time. Then nodes may fail after a while and need to be replaced by new ones. If there is no central administration of the network, then the IDs might be increased with each new sensor node or older ones might be reused several times. So what if we restrict our reasoning to algorithms which do not store or use node IDs?

We conjecture that this is the point where flooding becomes impossible.

CONJECTURE 9. *If node IDs cannot be stored nor passed along in the message, then no (asynchronous) flooding algorithm exists which obeys all Requirements 1 and 2 under Conditions 1 and 2.*

5. ROUTING WITHOUT IDENTIFIERS

Since flooding without identifiers appears impossible, one can imagine that a related problem, that of routing, is solvable within our constraints. That is, as long as one is sure that the destination exists, otherwise the problem degenerates to flooding. We will use the destination ID synonymously with a node being able to check locally whether it is the destination or not.

In this section, we will take a closer look at the routing problem of delivering a message from a source node s to a designated destination t. The idea for the algorithm is to estimate some finite upper bound $\hat{n} \geq n$ and then use the ideas from COUNTERFLOODING to guarantee correctness and termination. As we have seen, the difficulty lies in estimating the number of nodes without the use of IDs. In the routing case, this hurdle is overcome by the fact that the destination node t can acknowledge the receipt of the message and initiate a termination phase which will make use of the upper bound of the message counter.

Algorithm 4 Mobile Routing Algorithm at node v

Input: none
Events: receipt of message, $\Delta N \neq \emptyset$

mode = INIT

```
1: receive msg with n'
2: n̂ ← n' + 1
3: if (msg = FLOOD ∧ v = dest) ∨ (msg = TERM) then
4:    mode ← TERM
5: else
6:    broadcast FLOOD (message, n̂)
7:    mode ← FLOOD
8: end if
```

mode = FLOOD

```
1: if ΔN ≠ ∅ then
2:    n̂ ← n̂ + 1
3:    broadcast FLOOD (message, n̂)
4: end if
5: receive msg with n'
6: if msg = FLOOD ∧ n' > n̂ then
7:    n̂ ← n'
8:    broadcast FLOOD (message, n̂)
9: else if msg = TERM then
10:   n̂ ← max(n̂, n')
11:   mode ← TERM
12: end if
```

mode = TERM

```
1: process/delete message locally
2: CounterFlooding (n̂), msg = TERM (n̂)
3: receive msg with n'
4: if n' > n̂ then
5:    n̂ ← n'
6:    CounterFlooding (n̂), msg = TERM (n̂)
7: end if
```

The algorithm works as follows. Every node v stores a counter $\hat{n}$. Initially, every node is in INIT mode, that is, idle. Once a node has seen the message, it can be in one of two states: Either it assumes that the destination has not been reached and the message needs to be propagated at every opportunity, or else it *knows* that the destination has been reached and it needs to be careful about letting the other nodes know that they can stop soon as well so as to not endanger the termination requirement. The counter $\hat{n}$ is incremented in the first state and then used as an upper bound on $|V| = n$ in the second state. Once $\hat{n} \geq n$, we know that the CounterFlooding algorithm will guarantee us termination and correctness.

The details are given in Algorithm 4. The source s starts the algorithm by broadcasting FLOOD (message, 1).

Lemma 10. *Algorithm 4 is correct.*

Proof. Correctness is easily seen from the algorithm as nodes will continue sending (Line 6 in INIT and Lines 1-3 in FLOOD mode) until the destination has received the message (Line 3 in INIT) and will announce this with the acknowledgement TERM (Line 2 in TERM). □

Corollary 11. *Algorithm 4 achieves correctness after time at most $2Tn$ where T is the maximum message transmission time.*

Lemma 12. *If G keeps changing such that $\Delta N(F) \neq \emptyset$ for the set of nodes $F \subset V$ with mode(v) = FLOOD $\forall v \in F$, then any $x \in F$ will eventually enter TERM mode.*

Proof. Consider the terminated and flooding nodes, $T_t = \{v \in V \,|\, \text{mode}_t(v) = \text{TERM}\}$ and $F_t = \{v \in V \,|\, \text{mode}_t(v) = \text{FLOOD}\}$, respectively, depending on the time t. Initially, T_0 will contain the destination node (set $t = 0$ when the message has reached its destination).

If $\Delta N(v) \neq \emptyset$ for a node $v \in F_t$, this will increment $\hat{n}(v)$. If $\hat{n}(v) > \hat{n}(w)$ for $w \in N(v)$, then $\hat{n}(w) \leftarrow \hat{n}(v)$ after time $2T$. Thus, the highest estimate $\hat{n}$ will propagate throughout a connected component of flooding nodes.

Now assume there is a time t where all nodes $x \in T_t$ have stopped sending upon $\Delta N(x) \neq \emptyset$ (i.e., the call to CounterFlooding in Line 2/6 has finished) and $F_{t+2T} \neq \emptyset$. Then $\max_{x \in T_t} \hat{n}(x) < n$, otherwise the call to CounterFlooding in Line 6 would spread to all nodes in V. The eventual graph changes will now continually cause an increase of $\hat{n}(v)$ at some $v \in F_t$. At some time $\tilde{t} > t$, if not $F_{\tilde{t}} = \emptyset$, there will be $\hat{n}_{\tilde{t}}(v) \geq n = |V|$ at some $v \in F_{\tilde{t}}$. It will take at most $|F_{\tilde{t}}| < n$ hops for the message to reach a node $x \in T_{\tilde{t}}$ at time less than $\tilde{t} + 2Tn$. Then $n' = \hat{n}_{\tilde{t}}(v) > \hat{n}(x)$ which initiates the call to CounterFlooding with parameter at least n. Therefore, at some time $\hat{t} \leq \tilde{t} + 4Tn$, all nodes will have received a TERM message. □

Lemma 13. *Algorithm 4 terminates.*

Proof. By the definition of termination, we only have to prove that if the graph changes, the algorithm will stop sending messages eventually. Therefore, assume subsequently that G keeps changing. If the edge changes only involve the neighborhoods of nodes which have already TERM-inated, then Algorithm 4 will eventually stop sending messages because of a finite maximum $\hat{n}$ among the terminated nodes. Otherwise, by Lemma 12, we know that every node which has entered the FLOOD mode must eventually receive a TERM message. At that point, the highest estimate $\hat{N}$ is fixed since only FLOOD-ing nodes cause an increment of the counter (see Line 2 of FLOOD mode). Then $\hat{N}$ will be some finite value and all other nodes will update their $\hat{n} \leftarrow \hat{N}$ and the TERM messages will be propagated at most that many times to already terminated nodes (if the graph continues to be mobile). In other words, once all the nodes are TERM-inated, then they execute the CounterFlooding algorithm which we know will end for a finite $\hat{N}$. □

Lemma 14. *Algorithm 4 needs only $O(\log n)$ bits of storage and header size.*

Proof. In order to ensure small local storage and header size, we need to bound the largest estimate $\hat{n}$ by a polynomial in n. To that end, we examine the counter values of the above termination argument in more detail. We will keep track of the largest counter $\hat{n}$ in N.

Since the destination is reached after at most $2Tn$ time units, $N \leq 2n$ when the destination enters the TERM mode.

Thereafter, we have seen in Lemma 12 that once any node has an estimate $\hat{n} = n$ due to continual increases in its neighborhood, the rest of the algorithm will take its course to ensure that the nodes terminate. In the worst case, this information needs to travel less than n hops to a TERM node, so $N < 4n$ in the meantime. Once the call to CounterFlooding in Line 6 (TERM mode) is issued, it takes less than $2Tn$

time for all nodes to TERM-inate, which implies less than $2n$ more neighborhood changes. In the end, $N < 4n + 2n = 6n$ and $\hat{n}$ will not be incremented anymore. □

THEOREM 15. *The asynchronous, dynamic routing algorithm in Algorithm 4 fulfills all appropriate Requirements 1 and 2 under Conditions 1 and 2.*

Observe that while correctness is achieved in time $O(Tn)$, it might take a long time before the algorithm actually terminates at all nodes. Note that this makes this routing algorithms faster in reaching the designated destination than the other given algorithms except COUNTERFLOODING.

6. OPEN QUESTIONS

Altogether, the spirit of the paper is that finding fundamental limits and possibilities of mobile ad hoc networks is a goal well within reach. As this is only a first step, there remain numerous open questions.

Obviously, proving our conjecture about flooding in mobile networks is the foremost part of our future work. Another interesting aspect is that of allowing the use of identifiers which are small but not unique.

What about explicit termination? In other words, a node should *know* when all other nodes in the network have received the message. With the current (asynchronous) algorithms, this is not given if the graph does not change anymore at some point. In fact, this seems a costly requirement, and we suspect that it might not be possible in an asynchronous environment without further strong assumptions.

Also, we need to look at nodes joining and leaving during execution, that is, dynamic nodes as well as dynamic links.

7. REFERENCES

[1] Baruch Awerbuch. Reducing complexities of distributed magimum flow and breadth-first search algorithms by means of network synchronization. *Networks*, 15:425–437, 1985.

[2] Henri Dubois-Ferriere, Matthias Grossglauser, and Martin Vetterli. Age Matters: Efficient Route Discovery in Mobile Ad Hoc Networks Using Encounter Ages. In *Proceedings of the 4th ACM International Symposium on Mobile Ad Hoc Networking and Computing (MobiHoc)*, 2004.

[3] Faith Fich and Eric Ruppert. Hundreds of impossibility results for distributed computing. *Distributed Computing*, 16(2-3), 2003.

[4] Eli Gafni and Dimitri Bertsekas. Distributed algorithms for generating loop-free routes in networks with frequently changing topology. *IEEE Transactions on Communications*, 29(1), January 1981.

[5] Zygmunt J. Haas, Marc R. Pearlman, and Prince Samar. ZRP: A Hybrid Framework for Routing in Ad Hoc Networks. In *Ad Hoc Networking*. Addison-Wesley, 2001.

[6] Marc Heissenbüttel. *Networking in Ad-hoc Networks*. PhD thesis, University of Bern, CH-3012 Bern, Switzerland, June 2005.

[7] David B. Johnson, David A. Maltz, and Josh Broch. DSR: The Dynamic Source Routing Protocol for Multi-Hop Wireless Ad Hoc Networks. In *Ad Hoc Networking*. Addison-Wesley, 2001.

[8] Fabian Kuhn, Roger Wattenhofer, Yan Zhang, and Aaron Zollinger. Geometric ad-hoc routing: Of theory and practice. In *Proceedings of the 22nd ACM Symposium on Principles of Distributed Computing (PODC)*, 2003.

[9] Daniel Lang. A comprehensive overview about selected ad hoc networking routing protocols. Technical report, Technische Universität München, Department of Computer Science, March 2003.

[10] Moni Naor and Larry Stockmeyer. What can be computed locally? In *Proceedings of the 25th Annual ACM Symposium on Theory of Computing (STOC)*, 1993.

[11] David Peleg. *Distributed Computing: A Locality-Sensitive Approach*. Society for Industrial and Applied Mathematics (SIAM), 2000.

[12] Charles E. Perkins and Elizabeth M. Royer. Ad hoc On-Demand Distance Vector Routing. In *Proceedings of the 2nd IEEE Workshop on Mobile Computing Systems and Applications (WMCSA)*, 1999.

[13] Prince Samar, Marc R. Pearlman, and Zygmunt J. Haas. Independent Zone Routing: An Adaptive Hybrid Routing Framework for Ad Hoc Wireless Networks. *IEEE/ACM Transactions on Networking (TON)*, 12(4), 2004.