Rheinisch-Westfälische Technische Hochschule Aachen

Lehrstuhl für Informatik IV
Prof. Dr. rer. nat. Otto Spaniol

Efficient Flooding in Ad Hoc Networks

Seminar: Pervasive Computing
SS 2004

Frank Radmacher
Matrikelnummer: 235573

Betreuung: Stefan Penz
Lehrstuhl für Informatik IV, RWTH Aachen

Contents

1 Introduction

This essay deals with the *Broadcast Storm Problem* in *mobile ad hoc networks* and derives a framework for suitable solutions to this problem.

This essay is organised as follows: The first section gives an introduction to *mobile ad hoc networks*, their applications, and the importance of *efficient flooding* in these. Section 2 will clarify fundamental terms and techniques. The so called *Broadcast Storm Problem* will be explained in Section 3 in detail. Section 4 provides a generic framework for a suitable solution which is approved by simulation results in Section 5.

1.1 Mobile Ad Hoc Networks (MANETs)

A *mobile ad hoc network* (MANET) consists of wireless mobile hosts which form a temporary network without the aid of established infrastructure (i. e. base stations) or centralised administration (i. e. mobile switching centers). A MANET distinguishes itself from a traditional *wireless network* by its dynamic topology, no base station support, and its *multi-hop* communication capability. Every host has a limited transmission range and it cannot be guaranteed that every host can communicate with any other host directly in a *single-hop* fashion. Instead we have a *multi-hop* scenario, where packets from a source mostly have to be forwarded by other nodes to their destination. Beside this all hosts are allowed to roam freely in the network. Hosts can connect and disconnect to the network spontaneously. Power consumption could also be a critical issue (which isn't addressed in this essay). Thus a MANET is a fast deployable self-configuring wireless network characterised by unreliable media, node mobility, and dynamic topology structure.

Due to its on-the-fly construction, MANETs will be helpful in a wide range of applications, where networks need to be deployed immediately but base stations or fixed network infrastructures are not available. The variety range from scientific use (sensor networks, archaeological or ecological expeditions) over civilian use (disaster recovery, search and rescue) to military use (battlefield).

1.2 Broadcasts in MANETs

In this essay we examine how to perform a *broadcast* in an ad hoc network, i. e. the delivery of a packet to all other hosts in the ad hoc network. As described in [1], broadcasting is a common operation in MANETs. The crux of the matter is that broadcasting is widely used for solving network management problems. For example this can be sending an alarm signal, paging a particular host or finding a route to a particular host. Many routing protocols use broadcasts to exchange routing information. The network management problems solved by broadcasting covers also control messages, address resolution, as well as a possible last resort realisation of uni- and multicast messages in networks with a rapidly changing topology. However, due to the dynamic topology in MANETs, we expect broadcasts to occur more frequently.

2 Fundamentals

In the following we will explain fundamental terms and techniques required for the understanding of the following.

2.1 Broadcasting, Flooding, and Efficient Flooding

These terms are often used synonymously. To avoid confusion with these terms we will point out the differences in a few words. According to [2] sending a packet to all destinations in the network simultaneously is called *broadcasting*. In static networks *flooding* refers to a static routing algorithm, in which every incoming packet is sent on every outgoing line except the one it arrived on. Analogous in a MANET flooding just means that every host retransmits a received message once. Thus flooding is a possible simple realisation for a broadcast. We will use the terms *flooding* and *simple flooding* synonymously. With the term *efficient flooding* we refer more efficient algorithms than flooding which are able to realise a broadcast in a MANET.

2.2 The IEEE 802.11 MAC Sublayer Protocol

In the 802.11 MAC specification two modes of operation are supported. The first, called *Distributed Coordination Function* (DCF), does not use any kind of central control. The other, called *Point Coordination Function* (PCF), uses a base station to control all activity of radio communication. All implementations of IEEE 802.11 must support DCF but PCF is optional. An closer description of these operation modes can be found in [2]. Due to the fact that base station support is optional in MANETs, we assume the use of DCF in this essay.

When DCF is set up, a protocol named CSMA/CA which is the abbreviation for *Carrier Sense Multiple Access with Collision Avoidance* is used. CSMA/CA provides two operating methods. In the first method, when a station wants to transmit, it senses the channel. If it is idle, it just starts transmitting. It does not sense the channel while transmitting and emits its entire frame which may well be destroyed at the receiver due to interference there. If the channel is busy, the sender waits until the channel is idle using a *backoff algorithm* and then retries to start the transmission. More precisely, right after a host transmitted a message or when a host wants to transmit but the medium is busy and the previous backoff has been done, the CSMA/CA mechanism starts the backoff algorithm. A *backoff algorithm* forces a host to wait a random amount of time until it can retry starting the transmission. This is a commonly used mechanism for collision avoidance.

The other mode of the CSMA/CA protocol extends the first using virtual channel sensing realised by a RTS/CTS dialogue. Generally, in this mode the communication between two hosts is initiated by a *ready to send* (RTS) of sender and a *clear to send* (CTS) of the receiver. This is a common mechanism to avoid the *hidden station problem*, i. e. that a sent message interferes at the receiver.

However, due to the network overhead, usually only the first method is used for broadcasts. Hence in this essay we assume a CSMA/CA style communication without a RTS/CTS dialogue. Also consider that with respect to network overhead no acknowledgement mechanisms will be used for broadcasts.

2.3 Graph model

In this essay we mainly focus on a graph model to abstract MANETs, especially in Section 4. We abstract a MANET to a graph $G = (V, E)$ with a finite non-empty set of vertices V and a set of edges $E := \{\{u, v\} \mid u, v \in V,\ u \neq v\}$. The vertices or nodes represent the hosts. The edges stand for *bidirectional links* between hosts, i. e. the two hosts are in transmission range of each other. In this essay the terms *vertex*, *node*, and *host* as well as the terms *edge* and *link* are used synonymously.

In a wireless environment some links may be unidirectional, because the transmission range of two hosts may be different. However, those unidirectional links can be hidden to the network layer protocol using acknowledgement packets or unidirectional link detection mechanisms. Therefore, we assume all links are bidirectional, i. e. if node v can communicate with node u, then node u can also communicate with node v.

Always bear in mind, because of the dynamic topology of a MANET, the graph may change in time.

2.4 Graph theory related terms

We call $N(v) := \{u \in V \mid \{v,u\} \in E\}$ the *(1-hop) neighbor set* of $v \in V$. Further for $k \geq 2$ we denote $N_k(v) := N_{k-1}(v) \;\cup\; \bigcup_{u \in N_{k-1}(v)} N(u)$ as the *k-hop neighbor set* of $v \in V$.

The *degree* of vertex v is defined as $\deg(v) := |N(v)|$. In a graph $G = (V,E)$ the *maximum vertex degree* is denoted as $\Delta := \max_{v \in V} |\deg(v)|$.

A set $C(v)$ is called a *coverage set* of v if the neighbor set of v can be *covered* by nodes in $C(v)$, i. e. $N(v) - C(v) \;\subseteq\; \bigcup_{u \in C(v)} N(u)$.

We call a tuple of nodes $(u, v_1, v_2, \ldots, v_n, w)$ a *path* or *u-w-path* if $\{u, v_1\} \in E, \{v_1, v_2\} \in E, \ldots, \{v_n, w\} \in E$. The nodes $v_1 \ldots v_n$ are called *intermediate nodes* of the path.

In Section 4 the term *replacement path* for a node v connecting nodes u and w is used, which denotes formally the same as a u-w-path, but indicates that the node v isn't used as an intermediate node.

We call a set V of nodes *connected* if a u-v-path exists between every pair of nodes u and $v \in V$.

A set U of nodes is referred to as a *dominating set* of a graph $G = (V,E)$ if every node is either in the set or has a neighbor in the set, i. e. it holds $V - U \;\subseteq\; \bigcup_{u \in U} N(u)$. If U is connected, U is named a *connected dominating set* of G.

3 The Broadcast Storm Problem

To describe the *Broadcast Storm Problem* we first give a review of the basic functionality and general characterisation of broadcasting in a MANET. Then we give an overview about the occurring problems, followed by a detailed analysis of the individual problems.

3.1 Characteristics of Broadcasting in a MANET

We assume that every host is equipped with a CSMA/CA transceiver. A host can only send messages to hosts in its transmission range and the other way round a particular host can only receive messages from hosts which can reach the particular host with their transmission range.

A *broadcast* is a message to all other hosts in the network which has the characteristics as described in [1, 3]:

- **The broadcast is spontaneous.** Any mobile host can perform a broadcast at any time. This means that any kind of global synchronisation mechanism for broadcasts isn't possible. Because of node mobility and the dynamic topology, no global connectivity information can be obtained in advance. But little local and temporary connectivity information can be collected considering the necessary packet overhead carefully.

- **The broadcast is unreliable.** No acknowledgement mechanism will be used as mentioned in section 2.2. This means that it is acceptable if in some unlucky situations a broadcast gets lost or misses a host. For most applications like routing or address resolution an unreliable broadcast is sufficient. So in common 100% reliable broadcasts are unnecessary or can be realised at *application layer*.

Furthermore we assume that a host can detect duplicate packets and distinguish different broadcasts. This can be easily realised by attaching a source identifier and a sequence number to every message.

3.2 Problem Overview

Assume we perform a broadcast by *simple flooding*. Every host receiving the message the first time has to retransmit it. So we have n transmissions in a MANET consisting of n hosts. According to [1] this leads to the following problems:

Redundancy: When a mobile host retransmits a broadcast message, all its neighbors might already have received this message. Thus the bandwidth of the network gets reduced by unnecessary broadcasts.

Contention: Due to the fact that neighboring hosts often receive a rebroadcasted message nearly the same time, they all decide to retransmit the message nearly the same time. These messages very likely *compete* with each other. Thus the hosts have to wait for each other and congest the network. This phenomena is called *contention*.

Collision: Broadcast messages are rather sent simultaneously, such that *collisions* get more probable. Because of the lack of RTS/CTS dialogues, especially the *hidden station problem* is more likely to occur. Due to the absence of collision detection, collisions are more likely to cause more damage.

3.3 Analysis of Redundancy

First we discuss the redundancy of MANETs by two small examples. In Fig. 1(a) only two transmissions are necessary for a broadcast from the white

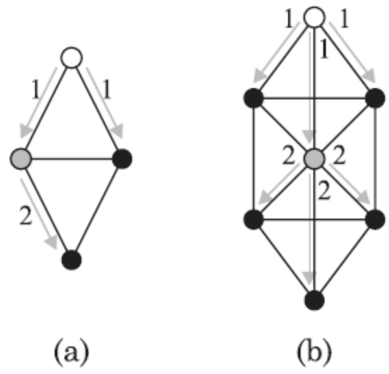


Figure 1: Two optimal broadcasting schemes. White nodes are sources, gray nodes retransmit the broadcast.

node, whereas four transmissions will be done by simple flooding. This means we can save 50% of transmissions in contrast to flooding. Fig. 1(b) shows an even more serious scenario: two transmissions are sufficient, so we can save five transmissions compared to flooding. Thus we can even save $\frac{5}{7}$ which are about 70% compared to flooding.

To give a more general analysis we consider the situation in Fig. 2, where host A sends out a broadcast message and host B decides to rebroadcast the message. Assume there currently are no other hosts rebroadcasting the message nearby B. Let the hosts have the transmission range r and the distance d to each other. Let S_A and S_B denote the circle covered areas of A and B, and further $S_{B-A} = S_B - S_{A \cap B}$ denotes the shaded region called the *additional coverage* of B. Then the additional coverage of B is $\pi r^2 - \mathrm{INTC}(d)$, where $\mathrm{INTC}(d)$ is the intersection of the two circles:

$$\mathrm{INTC}(d) = 4 \int_{d/2}^{r} \sqrt{r^2 - x^2} dx$$

When $d = r$, we get the upper limit of the additional coverage a host can gain, which equals $\pi r^2 - \mathrm{INTC}(d) = r^2(\frac{\pi}{3} + \frac{\sqrt{3}}{2}) \approx 0.61 \pi r^2$. Thus the additional coverage gained by a node can only be up to 61% in best case.

Even more interesting is the *average additional coverage* of a node. Assume B can be located arbitrarily in A's transmission range. So we obtain the average additional coverage by integrating $\pi r^2 - \mathrm{INTC}(d)$ over a circle of radius x centered at A for $x \in [0, r]$ and dividing this by the whole area covered by A:

$$\int_0^r \frac{2\pi x \cdot [\pi r^2 - \mathrm{INTC}(x)]}{\pi r^2} dx \approx 0.41 \pi r^2$$

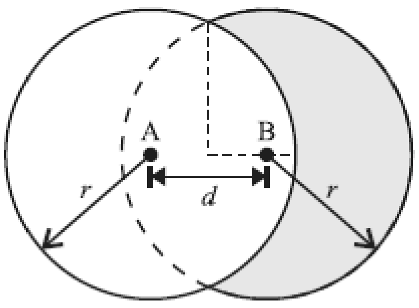


Figure 2: The gray shaded area visualises the additional coverage of node B which rebroadcasts a packet sent by A.

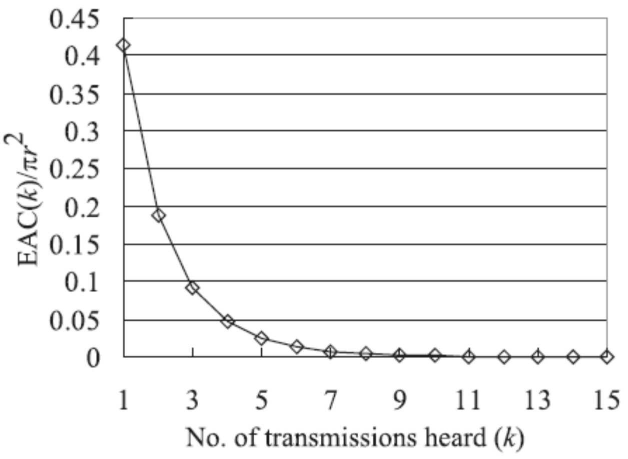


Figure 3: Analysis of redundancy: the expected additional coverage EAC(k) of a host after receiving a broadcast k times.

Thus the average additional coverage of a node which received a message for the first time is 41% of the area which is already covered.

Next we consider the scenario that a host received a broadcast message twice. If a host C receives a broadcast from hosts A and B the additional coverage of C is $S_{C-(A \cup B)}$. Through simulations with randomly generated positions of A and B in C's transmission range mentioned in [1], we obtain $|S_{C-(A \cup B)}| \approx 0.19\pi r^2$.

In general, we are interested in the *expected additional coverage* of a host after it heard the message k times. We will denote this function EAC(k). We know already EAC(1) $\approx 0.41\pi r^2$ and EAC(2) $\approx 0.19\pi r^2$. Like $k = 2$ the other values can also be obtained by simulation. The results from [1] are shown in Fig. 3. We see that for $k \geq 4$ the expected additional coverage is below 5%.

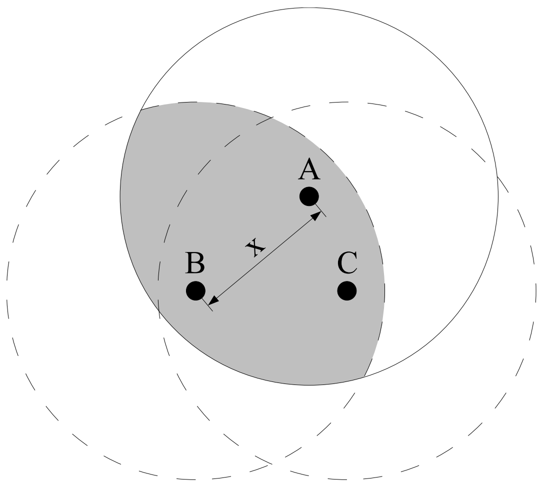


Figure 4: Host A sends a broadcast message to hosts B and C which compete with each other retransmitting A's message.

So we can conclude that most of the rebroadcasts of a node are superfluous in the case of simple flooding, especially when a node heard a broadcast several times.

3.4 Analysis of Contention

Consider a host transmits a broadcast message to n nearby other hosts. If these n hosts try to rebroadcast the message they likely *compete* with each other, i. e. while one of the n hosts rebroadcasts the message the other hosts *carrier sense* detect that the medium is busy. The second host can't retransmit his message until the first host finished his transmission and so on.

First we analyse the simpler case of $n = 2$. Assume a host A sends a broadcast message and according to Fig. 4 hosts B and C are receiving the message for retransmission. Let B randomly locate in A's transmission range. In order for C to compete with B, it must be located in the gray shaded area $S_{A\cap B}$. So the probability of contention is $|S_{A\cap B}|/\pi r^2$. Let x be the distance between A and B. We get the probability of contention by integrating $|S_{A\cap B}|/\pi r^2$ over the circle of radius x from 0 to r:

$$\int_0^r \frac{2\pi x \cdot \mathrm{INTC}(x)/(\pi r^2)}{\pi r^2} dx \approx 59\%$$

Of cause the expected contention will be higher with increasing n. For a more general framework we are interested in the probability $cf(n, k)$ that

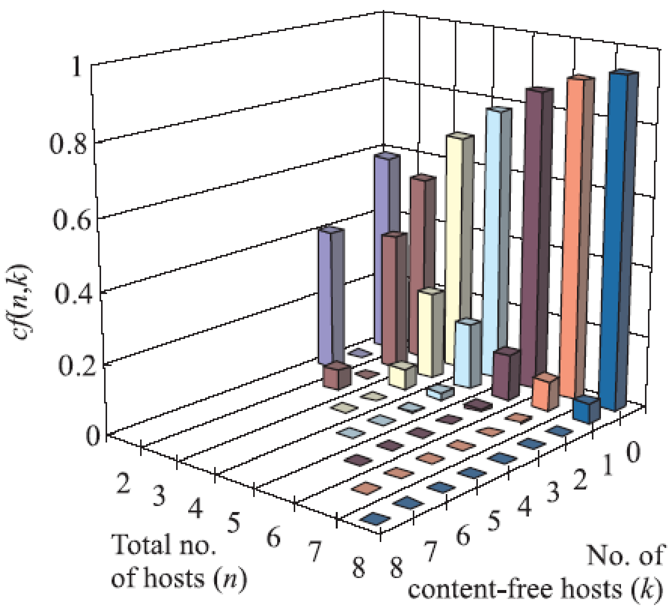


Figure 5: Analysis of contention: the probability $cf(n, k)$ of having k contention-free host among n receiving hosts.

k hosts out of our n hosts are *contention-free*. For example we already have derived that $cf(2, 0) \approx 0.59$. In [1] the values $cf(n, k)$ are obtained by simulation. The results can be seen in Fig. 5.

First we remark having $k = n - 1$ contention-free hosts implies having n contention-free hosts. In the case of $k = n - 1$ contention-free hosts there is no other host with which the n-th host could compete with. Hence the n-th host is also contention-free. So $cf(n, n - 1) = 0$ for all $n \in \mathbb{N}$.

The main result is, that the probability $cf(n, 0)$ that contention occurs between all hosts increases quickly over 80% with $n \geq 6$. Furthermore it is very unlikely that for $n \geq 3$ we have more than 1 contention-free host, so $cf(n, k) < 0.1\%$ for $n \geq 3$ and $k \geq 2$.

So we see that contention is likely to occur, especially in dense networks where several hosts receive a broadcast rather simultaneously.

3.5 Analysis of Collision

Due to the absence of base stations or access points in a MANET, we exclude the use of *Point Coordination Function* (PCF) and therefore assume the use of the *Distributed Coordination Function* (DCF) of the IEEE 802.11 specification as described in Section 2.2.

Against this background consider several hosts deciding to rebroadcast a

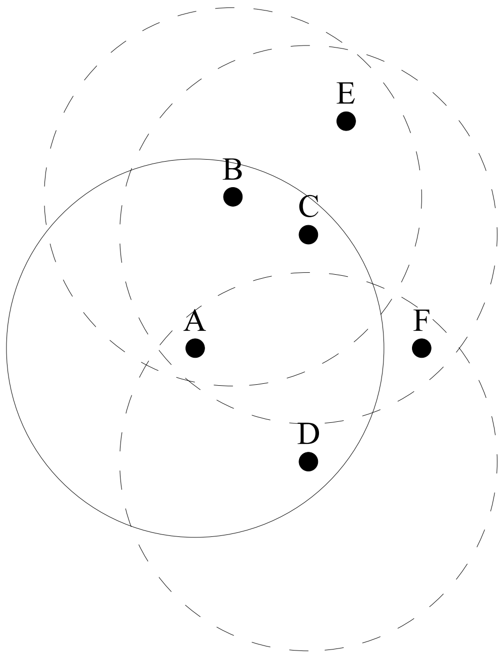


Figure 6: Host A sends a broadcast message. Host E can miss the broadcast due to collision of B and C. Host F can miss the broadcast due to the *hidden station problem* caused by C and D.

message heard from a host A as seen in Fig. 6. Then we have two reasons which make collisions likely to occur: First, after the medium of the receiving hosts has been quiet for certain time, all these hosts may have passed their backoff procedures. Thus it is possible that they retransmit the message at around the same time. Also consider carriers could not be sensed immediately due to processing delays and transmission latency. For example this could happen to hosts B and C under the aforementioned circumstances and hence host E could miss the broadcast. Second, because RTS/CTS dialogues are not used in broadcast messages, the *hidden station problem* may occur. For example hosts C and D can likely retransmit the broadcast message about the same time, because they are not able to detect their transmissions with carrier sense. Also in this example collision occurs and host F misses the broadcast.

Remember that no collision detection is available in wireless environments and in general no acknowledgement mechanism will be used for broadcasts. Thus collisions are a serious problem.

4 A generic approach based on Self-Pruning

In Section 3 we discussed the *broadcast storm problem* in MANETs. The analysed problems are redundancy, contention and collision which are all serious problems. Nevertheless all these problems have one cause in common, i. e. they increase with the number of hosts which unnecessarily rebroadcast a message. So the problems can be faced by the same approach which is to inhibit some nodes in the MANET from rebroadcasting.

Today, there are already many different techniques to face the broadcast storm problem. These techniques vary from basic heuristics like in [1, 3] to very explicit algorithms described and simulated in [4, 5]. Nevertheless, differences and advantages of these algorithms are often in implementation details and a general fair comparison of the underlying ideas is hard. Hence our scope is a more theoretical framework based on *self-pruning* according to [6].

4.1 Introduction to Self-Pruning

The approach of all mentioned papers to face the *broadcast storm problem* is to inhibit some nodes in the MANET from rebroadcasting. Therefore, the task is to select a subset of all nodes which retransmit broadcast messages. We will denote such a set as a *forward node set*. Basically, such a forward node set forms a *connected dominating set* (CDS). As defined in Section 2.4, a set of nodes is called a *dominating set* if every node is either in the set or has a neighbor in the set. Of course, in general an optimal solution will be a *minimum connected dominating set* (MCDS), i. e. a CDS with a minimal number of nodes. However, it has been proven that the MCDS problem is NP-complete. Nevertheless we should remember that we are in absence of global topology information and can only deal with local neighborhood information.

In [6] algorithms based on connectivity information of nearby nodes are called *neighbor-knowledge-based* algorithms, which can be further divided into *neighbor-designating* methods and *self-pruning*. In neighbor-designating methods which are sometimes also referred to as dominant-pruning [7, 8], the forwarding status of each node is determined by its neighbors. In contrast to this approach, in self-pruning every nodes decides by itself to retransmit a broadcast or not. Because we want to focus on a generic framework, we will use the clearer approach of self-pruning. Anyway, the results of this study can be used for other variants.

First we discuss which topology information can be collected by a node v:

- First, *k-hop neighbor set information* $N_k(v)$ can be collected. This can be done by distributing the neighbor set $N(v)$ within $k-1$ hops. All nodes receiving this information get complete knowledge of all links within $k-1$ hops and partial connection information of nodes in k hops. In fact only connections between nodes in $k-1$ hops and nodes in k hops are known at v.

- Second, every broadcast can carry a *routing history* with it, i. e. a list of the most recent nodes, which have already retransmitted the broadcast. We will denote this *routing history* as *visited nodes set* $D_h(v)$. The index h means that the set contains at most the last h visited nodes.

Furthermore we can assume that every node v is able to establish an unambiguous order of the collected nodes in $N_k(v)$ and $D_h(v)$, i. e. every node has an unique *priority value*. The simplest way to achieve this is by sorting nodes by their unique *node id*. Other possibilities are *node degree*, i. e. the number of neighbors of a node, and *neighborhood connectivity* which is defined as the ratio of pairs of not directly connected neighbors to pairs of any neighbors. Anyway, node ids are still necessary to make priority values unambiguous.

In the following we will present several *coverage conditions* with which a node can decide its forwarding status. Simulation results in [6] show that the presented coverage conditions are a good approximation for the MCDS. Note that the coverage conditions presented in Section 4.2 and 4.3 make use of global neighborhood information in their worst cast, and therefore are not suitable for implementation. But they lead to the generic self-pruning scheme presented in Section 4.4.

4.2 Self-Pruning with static Coverage Conditions

First we will present static conditions with which a node can decide its forwarding status. *Static* means that no routing history is used and thus the obtained CDS is independent of the source of the broadcast.

The first coverage condition based on static neighborhood information checks the necessity of forwarding for a node v by the existence of *replacement paths* for v. A *replacement path* for v that connects v's neighbors u and w is a u-w-path via other intermediate nodes than v. Thus all paths which connect u and w except the path (u, v, w) are replacement paths for v. For example in Fig. 7 $(u, 5, 6, w)$ or $(u, 9, 7, 6, w)$ are such replacement paths for v. If for a node v any pair of neighbors u and w can be connected via a *replacement path*, v can get a non-forward status, because every of v's neighbors can be reached via other nodes. Furthermore a priority check is built into the coverage condition, so that two nodes don't decide independently that a replacement path through the other node exists and thus no node sees the requirement for a forwarding.

> **Coverage Condition I (static):**
> Node v has a non-forward node status if for any two neighbors u and w, a *replacement path* exists that connects u and w via several intermediate nodes (if any) with higher priority values than the priority value of v.

It is obvious that the vertex subset derived by applying coverage condition I forms a CDS, because for every node v with non-forward node status is ensured that it is reachable from all nodes in $N(v)$ via a replacement path with higher node priority and thus there must be at least one forward node in $N(v)$. Also the CDS is connected, because for every non-forward node v is ensured that a replacement path for v exists which leads iteratively to a replacement path only consisting of forwards nodes which will be denoted as a *maximal replacement path*.

In order to see that such a *maximal replacement path* can be constructed, we describe an algorithm to check the existence of a replacement path. The following presented method finds a replacement path recursively. It constructs a *maximal replacement path*, such that all intermediate nodes are forward nodes. That is the case if none of the nodes in the maximal replacement path can be further replaced according to coverage condition I.

First we define two special kinds of nodes:

minimum node: In a path $P = (u, v_1, ..., v_n, w)$ a *minimum node* is the intermediate node v_i with lowest priority value.

max-min node: Assume $\{P_1, \ldots, P_n\}$ is the set of all replacement paths for node v that connect u and w. Then a *max-min node for* (u, w, v) is the node with the highest priority value of all *minimum nodes* in $P_1, \ldots, P_n$.

Next we define a procedure called MaxMin to construct a maximal replacement path for v that connects u and w:

```
MaxMin($u, w, v$)
1: if $u$ and $w$ are directly connected then return $\emptyset$.
2: Find the max-min node $x$ for $(u, w, v)$.
3: return path (MaxMin($u, x, v$), $x$, MaxMin($x, w, v$)).
```

The procedure MaxMin(u, w, v) will complete in a finite number of steps. The path $(u$,MaxMin$(u, w, v), w)$ is the demanded replacement path, whose nodes cannot be further replaced.

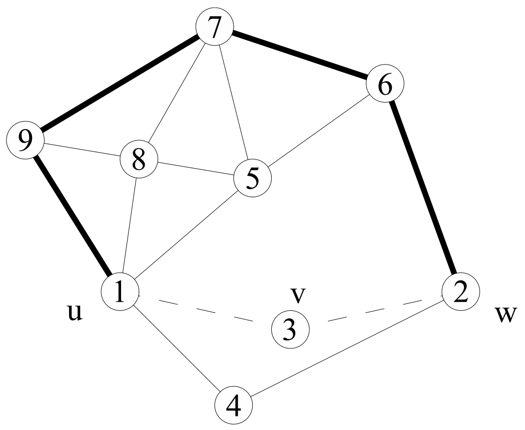


Figure 7: A sample maximal replacement path for v that connects u and w.

All nodes selected by $\textsc{MaxMin}(u, x, v)$ and $\textsc{MaxMin}(x, w, v)$ have higher priorities than x because x is a *max-min node* which has been selected before. No node can be selected twice as a *max-min node*: Assume a node x has been selected twice in two different recursive steps. Then the resulting replacement path can be further replaced by shorten the path at node x, but this contradicts to the fact that x is a *max-min node*. For the same reason a node x in a maximal replacement path cannot be further replaced: If x can be replaced by a path P, then $(\textsc{MaxMin}(u, x, v), P, (x, w, v))$ is another replacement path for v that connects u and w. Clearly, all the nodes in this path have a higher priority than x what contradicts to the fact that x is a *max-min node*.

Now we will demonstrate the $\textsc{MaxMin}$ algorithm on basis of the sample graph of Fig. 7. It shows a replacement path for v that connects u and w. The max-min node for (u, w, v) is 6, because all paths that connect u and w have 4, 5 or 6 as minimum node. The max-min node for $(u, 6, v)$ is 7, and the max-min node for $(u, 7, v)$ is 9. Therefore, the maximal replacement path is $(u, 9, 7, 6, w)$. Note that the priority values of u and w are of no importance for coverage condition I.

To check coverage condition I for a node v, each node needs to check every pair of its neighbors, and for every such pair all nodes with higher priority have to be checked. There are $\binom{\deg(v)}{2} \in O(\deg(v)^2)$ such pairs, so the overall computational complexity at each node in a graph with n nodes is $O(n\Delta^2)$, where Δ is the *maximum vertex degree* in the network as introduced in Section 2.4.

In order to reduce computational complexity at each node, we present a second coverage condition which allows a computation in $O(n\Delta)$. Coverage

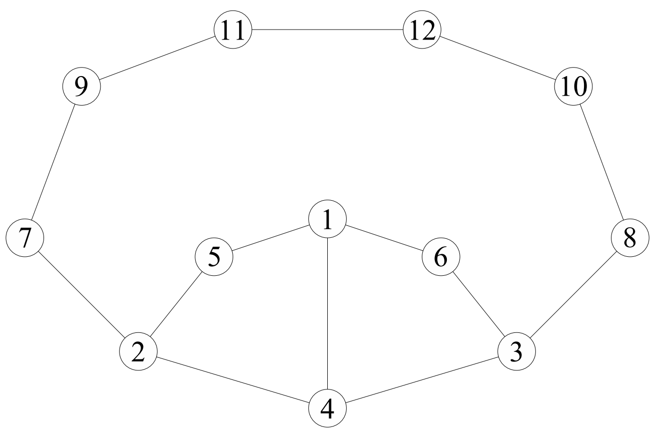


Figure 8: Node 4 satisfies coverage condition I but not coverage condition II.

condition II makes use of *coverage sets.* As defined in Section 2.4, a set $C(v)$ is called a *coverage set* of v if the neighbor set of v can be *covered* by nodes in $C(v)$. Each node determines the forward node status by checking that all neighbors are already *covered* by a connected set of nodes with higher priority.

> **Coverage Condition II (static):**
> Node v has a non-forward node status if it has a coverage set. In addition the coverage set belongs to a connected component of the subgraph induced from nodes with higher priority values than the priority value of v.

Coverage condition II is stronger than coverage condition I. When a node v satisfies coverage condition II, it also satisfies coverage condition I, because the existence of a *connected coverage set* implies the existence of a *replacement path* for any pair of v's neighbors. Thus also the vertex subset derived by applying coverage condition II forms a CDS.

But generally the reverse situation does not hold as shown in the sample graph of Fig. 8. There exists no connected coverage set consisting of nodes with higher priority values for node 4. In fact the node set $\{5, 6, 7, \ldots, 12\}$ is a coverage set of 4 consisting of nodes with higher priority than 4, but the node set is not connected. Contrariwise for every pair of 4's neighbors a replacement path exists consisting of nodes with higher priority values than 4. Thus node 4 satisfies coverage condition I, but not coverage condition II. However, simulation results in Section 5 will show that these two conditions are very close in reducing the number of forward nodes.

In the following we discuss an algorithmic approach to find a coverage set according to coverage condition II. Let v be the current node which has

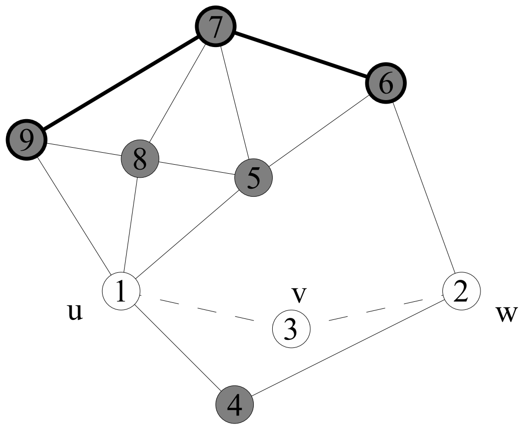


Figure 9: Sample coverage sets for v.

to decide its forwarding status, let $G'(v)$ be the subgraph of G induced from nodes with higher priority values than v, and n the number of nodes with higher priority values than v.

First decompose $G'(v)$ into connected components $V_1, V_2, \ldots, V_l$. This can be done via *depth-first search* in $O(n\Delta)$. Next compute for each component V_i the set of covered neighbors $N(V_i) := \bigcup_{w \in V_i} N(w)$, and check if there exists a V_i such that $N(v) \subseteq N(V_i)$. This can be done in $O(n\Delta)$. Thus the overall complexity is $O(n\Delta)$.

Again, we take a look at a sample graph. In Fig. 9 we see several connected coverage sets for v that satisfies coverage condition II. By the algorithmic approach we can decompose $G'(v) = \{4, \ldots, 9\}$ into connected components $V_1 = \{4\}$ and $V_2 = \{5, 6, 7, 8, 9\}$. The condition $N(v) = \{u, w\} \subseteq N(V_i)$ holds (even for both $i = 1$ and $i = 2$), therefore v has a coverage set. Of course there are also more coverage sets which are not covered by the algorithmic approach. In this example the maximal replacement path is a coverage set for v, too.

4.3 Self-Pruning with dynamic Coverage Conditions

In the static versions of the coverage conditions a node doesn't bother which nodes a broadcast message has already passed. So a node can decide to forward a message just to cover nodes which have already heard the message.

The static versions of the coverage conditions can be extended to the dynamic versions by taking advantage of the *routing history*. Therefor we assume that every broadcast message includes a list of *visited nodes* $D_h(v)$, i. e. the list of the last h nodes that have forwarded the broadcast message. Because visited nodes have already forwarded the broadcast, we can treat

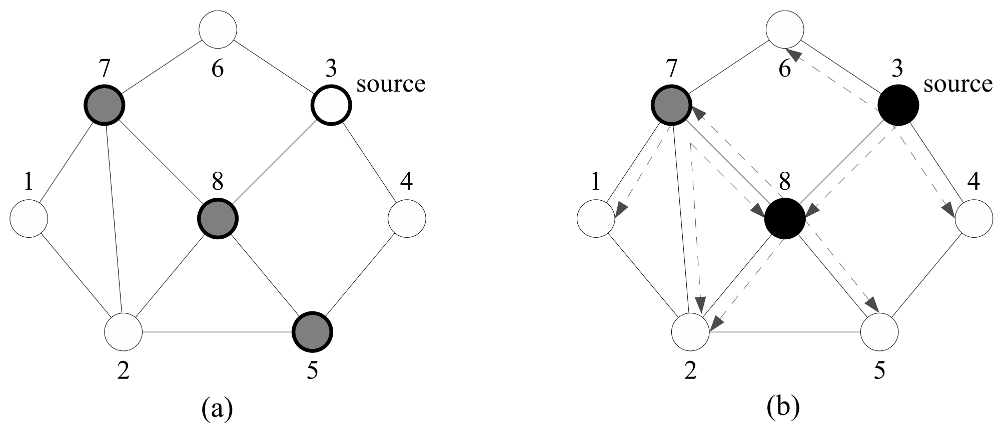


Figure 10: (a) Forward node set without routing history (static). (b) Forward node set with routing history (dynamic) with node 3 as source. Black nodes are visited nodes and gray nodes are forward nodes.

all visited nodes as nodes with forwarding status and hence as nodes with a higher priority than all non-forward nodes. Thus we can add this result to our coverage conditions. Note that the static conditions are special cases of the dynamic conditions, that is in the case of $h = 0$ and thus $D_h(v) = \emptyset$.

> **Coverage Condition I (dynamic):**
> Node v has a non-forward node status if for any two neighbors u and w, a *replacement path* exists that connects u and w via several intermediate nodes (if any) with higher priority values than the priority value of v or with visited node status in $D_h(v)$.
>
> **Coverage Condition II (dynamic):**
> Node v has a non-forward node status if it has a coverage set. In addition the coverage set belongs to a connected component of the subgraph induced from nodes with higher priority values than the priority value of v or from nodes with visited node status in $D_h(v)$.

In Fig. 10 two examples of forward node sets on the same network are shown: One without routing history (Fig. 10(a)) and one with routing history (Fig. 10(b)). In this example the use of both, coverage condition I and II, results in the same forward node set. Consider node 3 is the source of the broadcast and hence gets visited node status. After passing node 8, node 5 receives the broadcast message with its routing history. Thus it can conclude that it should get a non-forward node, because the neighbor set of node 5 can be covered by nodes 3 and 8. For the case that node 3 is the source of the broadcast the nodes which have to forward the package are bold bordered in Fig. 10. We see, that the number of those nodes could be reduced from 4 to 3 by using of the routing history.

Although the calculated forward node set with routing history is dependent of the source node, it can be used globally for all other broadcasts as long as the topology does not change. This is simply because a broadcast is supposed to reach all nodes in a network. Anyway, the forward node sets derived by the dynamic coverage conditions are advantageous for the source of a broadcast.

The complexity of the dynamic versions of the coverage conditions stays the same, because their underlying algorithms for decision are analogous to the static case. However, the routing history adds some additional network overhead to broadcast messages.

4.4 Self-Pruning with k-hop approximation Coverage Conditions

The above static and dynamic coverage condition have the disadvantage that in the worst case global topology information is needed to find a replacement path or a coverage set according to the respective coverage condition. But we already discussed in Section 3.1 that it is only feasible to collect small neighborhood information. Therefore, the above coverage conditions are not suitable for implementation, and we have to restrict them by using less neighborhood information.

In the following approach of *k-hop approximation* we assume that a node v only knows topology information of its *k-hop neighbor set* $N_k(v)$. In detail, according to the realisation described in Section 4.1, v knows all connections of nodes within $k-1$ hops, i. e. between all nodes in $N_{k-1}(v)$, but only partial information about connections in k hops. In fact only connections between nodes in $N_k(v) - N_{k-1}(v)$ and nodes in $N_{k-1}(v)$ are known at v. In the following we will implicitly assume that only these links between nodes are used, when we speak of replacement paths or connected sets in $N(v)$. With this in mind we can formulate our two coverage conditions using *k-hop approximation*.

> **Coverage Condition I (k-hop approximation):**
> Node v has a non-forward node status if for any two neighbors u and w, a *replacement path* exists that connects u and w via several intermediate nodes (if any) in $N_k(v)$ with higher priority values than the priority value of v or with visited node status in $D_h(v)$.

Coverage Condition II (k-hop approximation):
Node v has a non-forward node status if it has a coverage set. In addition the coverage set belongs to a connected component of the subgraph induced from nodes in $N_k(v)$ with higher priority values than the priority value of v or from nodes with visited node status in $D_h(v)$.

Because these generic coverage conditions are less complex in computation than the former, the complexity stays in $O(n\Delta^2)$ respectively $O(n\Delta)$. But now we can give more precise upper bounds which involve the parameter k. Considering the two parameters n and Δ, we can narrow $\Delta \leq \pi r^2 D = cD$ and $n \leq \pi(kr)^2 D = ck^2 D$, where c is a constant, r is the transmission range, and D is an upper bound for the *density* of the network, i. e. the number of nodes per unit area. Thus the complexity is $O(n\Delta^2) = O((ck^2D) \cdot (cD)^2) = O(k^2D^3)$ for coverage condition I and $O(n\Delta) = O((ck^2D) \cdot (cD)) = O(k^2D^2)$ for coverage condition II.

This shows, that not only a too large chosen parameter k but also a too dense network could cause unacceptable computational complexity in each node. Although appropriate density is necessary for network connectivity and redundancy, a very dense network is inefficient due to *contention* and *collision* as described in Section 3. However, as mentioned in [1, 6], there are *distance based* and *cluster based* approaches to alleviate this problem.

With *k-hop approximation* we gained a generic self-pruning approach with four parameters:

1. The parameter k restricts the neighborhood information which every node collects. Because a larger k-hop neighbor set $N_k(v)$ allows a better approximation of a MCDS, a higher k leads to a smaller CDS and thus to a smaller forward node set, but to higher network overhead due to exhausting collection of neighborhood information. Vice versa a smaller k leads to larger forward node sets, but less network overhead is needed for the computation of a CDS. Also the collection of k-hop neighborhood information for a large k requires more rounds of information exchange until the corresponding pieces of information are collected. Therefore, collection of k-hop neighborhood information takes a longer time for a large k because it needs k rounds of information exchange to obtain the k-hop neighbor set $N_k(v)$ after a topology change.

2. The parameter h with $h \leq k$ restricts the routing history which is added to broadcast messages. A higher h can reduce the forward node set, but leads to bigger broadcast messages. Vice versa, with a small

h the reduction of forward nodes may be suboptimal, but broadcast messages cause less network overhead. In the case $h = 0$ no routing history is used.

3. Of course we have the choice between *coverage condition I* and *coverage condition II*. As mentioned above these two conditions differ in the derived forward node set as well as in computational complexity.

4. Different functions to get the *priority values* can be used. Three common candidates are *node id*, *node degree*, i. e. the number of neighbors of a node, and *neighborhood connectivity*, which is defined as the ratio of the number of pairs of not directly connected neighbors to the number of pairs of any neighbors. A disadvantage of node degree and neighborhood connectivity is that node degree relies on 1-hop neighborhood information and neighborhood connectivity relies on 2-hop neighborhood information. So they cause an additional network and computation overhead. Also node degree and neighborhood connectivity of a node v may change in a dynamic network and thus the priority of v may change, too.

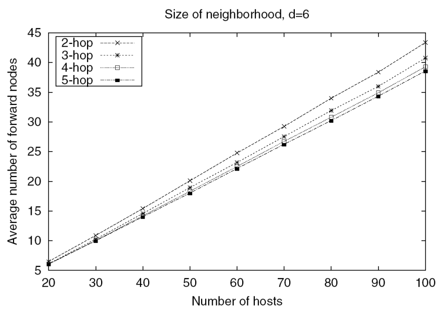
5 Simulation & Conclusion

This last section deals with simulation results from [6]. Several parameter settings are examined and an optimal configuration will be presented in the conclusion.

5.1 Simulation setup and parameters

The simulation results which we take from [6] have been achieved with their simulator *ds* [9]. It is worth to mention that *ds* only simulates functions in the network layer, assuming an ideal MAC layer without contention or collision. The setup parameters for the simulation are the number of hosts n and the average node degree d. The random ad hoc networks has been generated by placing n hosts randomly in a 100×100 area. The transmission range r has then been adjusted to produce $\frac{nd}{2}$ links. Networks which couldn't form a connected graph have been discarded. Note that node mobility has not been taken into consideration.

The parameters which have been changed for various simulations are the four parameters of the generic self-pruning scheme: (1) number of hops k for collecting neighborhood information, (2) length h of the routing history,



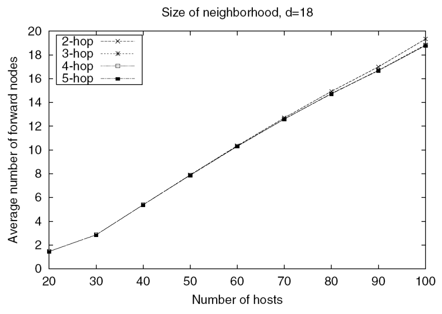
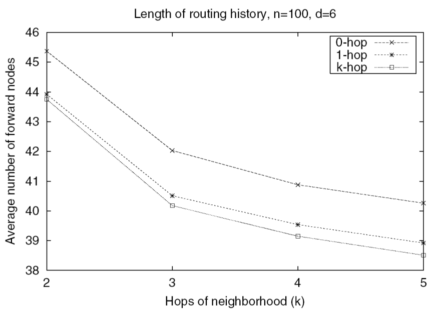


Figure 11: Efficiency of k-hop approximations of coverage conditions with various k's.



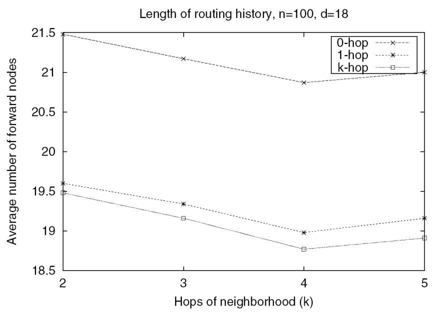


Figure 12: Efficiency of coverage conditions with various lengths h of routing history.

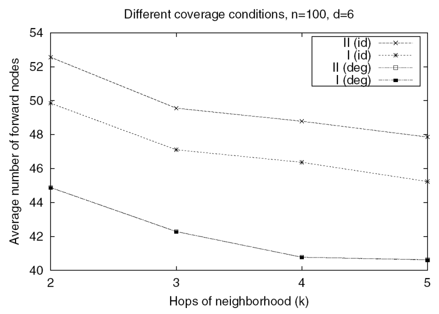
(3) type of coverage condition, and (4) the type of priority function. We have already described these parameters in Section 4.4 in detail.

The base configuration (if not annotated different) is the use of coverage condition I with 2-hop neighborhood information, 2-hop routing history, and node degree as priority function.

For all simulations two different settings for the density of the network have been used: The setting $d = 6$ gives us the circumstances of a relatively *sparse* network, the setting $d = 18$ represents a relatively *dense* network.

5.2 Simulation results

In Section 4.4 we have mentioned the disadvantages of large values for our first parameter, i. e. the number of hops k for collecting the neighborhood information. Simulation results for a varying k are shown in Fig. 11. Four configurations have been compared: $k = h = 2$ (2-hop), $k = h = 3$ (3-hop), $k = h = 4$ (4-hop), and $k = h = 5$ (5-hop). We see that even in this idealised simulation without contention, collision, and node mobility, the



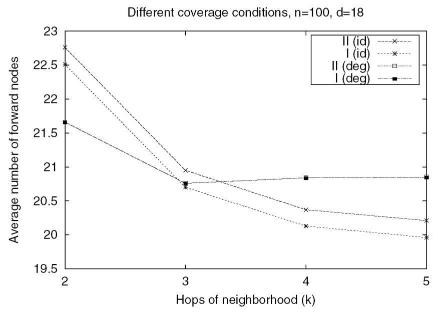
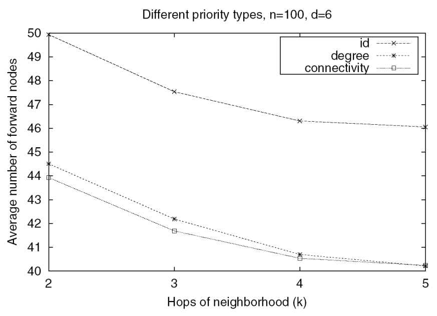


Figure 13: Efficiency of different coverage conditions.



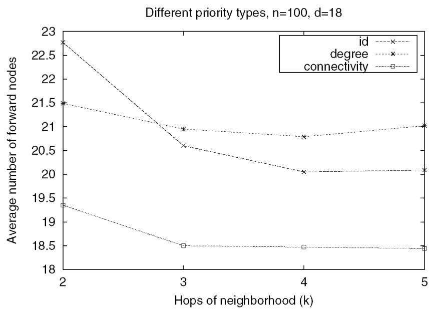


Figure 14: Efficiency of coverage conditions with various priority functions.

efficiency gain is minimal. With larger than 2-hop neighborhood information the average number of forward nodes can only be reduced by a few nodes in sparse networks, i. e. even in the case $n = 100$ 3-hop is about 10% less effective than 2-hop. In dense networks the gain of efficiency is even less. Thus we conclude that 3-hop neighborhood information is only acceptable for relatively sparse and static networks. In most cases 2-hop neighborhood information will be the best choice.

Concerning the length h of the routing history, simulation results for various lengths h are shown in Fig. 12. Three configurations have been compared: no routing history (0-hop), one hop routing history (1-hop), and the case $h = k$ (k-hop), i. e. full routing history in the known neighborhood. This simulation is conducted on networks with 100 nodes ($n = 100$), with k varying from 2 to 5. We see that 0-hop is about 5% less efficient than 1-hop in sparse networks and about 10% less effective in dense networks. The efficiency of k-hop is very similar to 1-hop. So we can conclude that the use of 1-hop routing history is the most cost-effective setting.

Simulation results for the type of coverage condition and the type of priority function are covered in Fig. 13 and 14. Fig. 13 compares the two coverage conditions with different priority functions. We see that I(id) is only slightly more efficient than II(id). Furthermore no differences can be observed between I(deg) and II(deg). We conclude, considering the computation overhead, coverage condition II is a good approximation of coverage condition I.

Fig. 14 compares three different priority functions. The results show that from a theoretical point of view the smallest forward node set can be achieved by neighborhood connectivity followed by node degree. In sparse networks, neighborhood connectivity is only slightly better than node degree. Node id leads usually to the largest forward node set. But we have to bear in mind that 1-hop neighborhood information is needed to compute the node degree and 2-hop neighborhood information is required for the computation of the neighborhood connectivity. So a choice of one of these priority functions only make sense if nodes collect 1-hop respectively 2-hop neighborhood information. Also we have to take into consideration the idealised conditions of our simulation especially the absence of node mobility. Node degree and neighborhood connectivity of a node can change frequently which can lead to wrong computations caused by outdated priority values. From this point of view it is difficult to make an optimal choice for the priority function. Neighborhood connectivity certainly achieves best results in MANETs with low node mobility, but probably in MANETs with a more frequently changing topology node id is a better choice.

We conclude that coverage condition II should be used with node id or neighborhood connectivity as priority function.

5.3 Conclusion

In this essay we characterised *mobile ad hoc networks* (MANETs) and pointed out the main problems. We discussed the *broadcast storm problem* and analysed the effect of *redundancy*, *contention*, and *collision*. These problems can be alleviated by allowing only a subset of hosts to rebroadcast messages. These hosts form a *forward node set* which should be a rather small *connected dominating set* (CDS). We attended the selection of the forward node set by a generic approach based on self-pruning. We derived two suitable *coverage conditions* for approximating a *minimum connected dominating set* (MCDS). These coverage conditions are scalable in computational complexity and network overhead by the choice of various parameters. Through simulation results we conclude that a good configuration of these parameters are the usage of coverage condition II with 2-hop neighborhood information, 1-hop routing history, and node id or neighborhood connectivity as priority function.

References

[1] Sze-Yao Ni, Yu-Chee Tseng, Yuh shyan Chen, and Jang-Ping Sheu. The broadcast storm problem in a mobile ad hoc network. *MobiCom*, 1999.

[2] Andrew S. Tanenbaum. *Computer Networks, Fourth Edition.* Prentice Hall PTR, 2002.

[3] Yu-Chee Tseng, Sze-Yao Ni, and En-Yu Shih. Adaptive approaches to relieving broadcast storms in a wireless multihop mobile ad hoc network. *Infocom*, 2001.

[4] Brad Williams and Tracy Camp. Comparison of broadcasting techniques for mobile ad hoc networks. *MOBIHOC*, 2002.

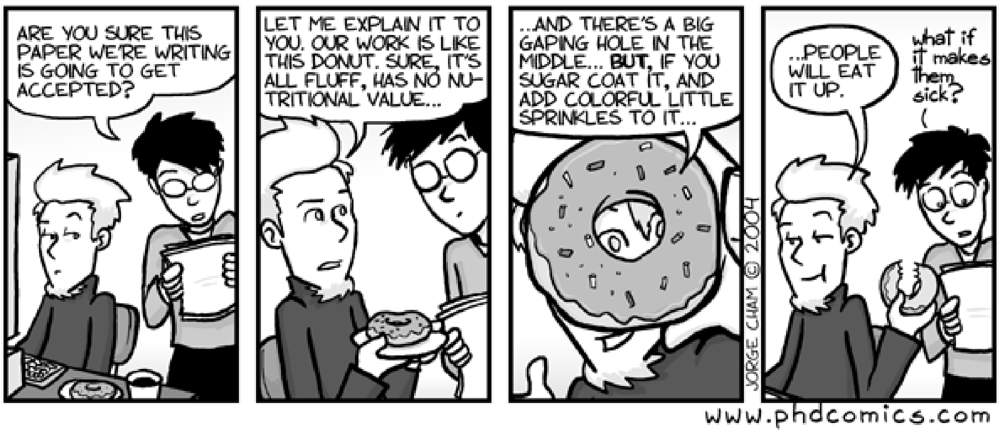
[5] Yunjung Yi, Mario Gerla, and Taek-Jin Kwon. Efficient flooding in ad hoc networks: a comparative performance study. *IEEE ICC*, 2003.

[6] Jie Wu and Fei Dai. Broadcasting in ad hoc networks based on self-pruning. *Infocom*, 2003.

[7] Hyojun Lim and Chongkwon Kim. Flooding in wireless ad hoc networks. *Computer Communications 24(3-4)*, 2001.

[8] Hyojun Lim and Chongkwon Kim. Multicast tree construction and flooding in wireless ad hoc networks. *ACM MSWiM*, 2000.

[9] Fei Dai. Wireless routing simulation suit (wrss). http://sourceforge.net/projects/wrss/, 2001.


ARE YOU SURE THIS PAPER WE'RE WRITING IS GOING TO GET ACCEPTED?
LET ME EXPLAIN IT TO YOU. OUR WORK IS LIKE THIS DONUT. SURE, IT'S ALL FLUFF, HAS NO NUTRITIONAL VALUE...
...AND THERE'S A BIG GAPING HOLE IN THE MIDDLE... BUT, IF YOU SUGAR COAT IT, AND ADD COLORFUL LITTLE SPRINKLES TO IT...
JORGE CHAM © 2004
...PEOPLE WILL EAT IT UP.
what if it makes them sick?
www.phdcomics.com