A Protocol for Data Dissemination in Frequently Partitioned Mobile Ad Hoc Networks

Jörg Hähner[*], Christian Becker, and Kurt Rothermel
University of Stuttgart
Institute of Parallel and Distributed Systems (IPVS)
Universitätsstraße 38
70569 Stuttgart, Germany
{haehner, becker, rothermel}@informatik.uni-stuttgart.de

Abstract

Distribution of data in mobile ad hoc networks is challenged when the mobility of nodes leads to frequent topology changes. Existing approaches so far address either the network partitioning problem or are capable of handling large amounts of data, but not both at the same time.
In this paper a novel approach is presented which is based on a negotiation scheme enhanced by an adaptive repetition strategy. Different strategies for the selection of repeated data are presented and evaluated. Simulation results show a reduction of data transfer volume compared to hyper-flooding by 30 to 40% even in the presence of frequent network partitions.

1. Introduction

Mobile ad hoc networks (MANETs) are going to be a reality in the near future with more and more mobile devices, e.g. PDAs or cell-phones, being equipped with short range radio technology, e.g. as Bluetooth or 802.11. In our daily environments such MANETs will not only contain the mobile nodes which are typically carried by their users but also incorporate devices being fixed in the infrastructure, such as sensors or information provision points, e.g. info-stations. Applications in such environments can make use of the information being available through the sensors and other nodes. Examples are tracking applications in production plants capturing the location of production material and the state of manufacturing machines, communication on a construction site, missions from civil services, e.g. collaborative fire-fighting, but also convenience applications such as smart city/shopping guides.

Typically, information in such networks itself is spatially scoped, i.e. only from interest within a distinct area nearby the information source. Sensor networks, i.e. ad hoc networks with typically stationary nodes, can setup links between information sources and sinks. Mobility challenges the information dissemination in such networks, since network partitions cannot be treated as errors because they happen regularly. In order to supply applications on nodes with information of their environment a robust mechanism to deliver data is needed. In order to increase availability of data, replication is a candidate to achieve this goal with a trade-off to consistency.

In this paper we present an algorithm for updating replicated data on mobile nodes which is gathered by information provided by sensors. We refer to such data as *model-data*, since the sensor information provides a model of real-world's state. The consistency of the replicated data is weak, due to unpredictable network partitioning, aiming at delivering the most current state of an information entity and not providing single-copy consistency. Current information shall replace older one and inconsistencies are tolerated as long as the most current information will finally be propagated. Using a hyper-flooding [OT98] appproach as the foundation of a three-way-handshake protocol enables our protocol to overcome network partitions. The negotiation of transferred data leads to a significant reduction of the data transfer volume compared to plain hyper-flooding by 30 to 40%.

Next we will introduce the system model. After a discussion of existing flooding techniques for data propagation in ad hoc networks our algorithm is described. Performance results from simulations are presented based on two scenarios before the discussion of related work and an outlook to future work concludes the paper.

2. System Model

The system consists of two kinds of nodes: observer nodes and mobile nodes. *Observer nodes* are equipped with a synchronized real-time clock (e.g. GPS clock) or an appropriate clock synchronisation algorithm [Roem01], and

[*] Jörg Hähner is funded by the Gottlieb Daimler and Carl Benz Foundation, Ladenburg, Germany

sensors allowing to make observations in their proximity that describe properties of the real world. Every *observation* represents a state change of an *object* that has a unique object ID (*oid*), and has a time to live (*TTL*) that depends on the type of observation. Each observation is timestamped with t_{obs} by the observer node to indicate when the observation was made. Additional information (*info*) may be added by the observer node to describe precisely what kind of state change was observed, e.g. the position (state change described in *info*) of a person (object) or the temperature inside a room. The tuple (*oid*, *TTL*, t_{obs}, *info*) is called *meta-data* because it describes the „what and when" of an observation. The actual distinction of objects on the sensor level is not part of this paper.

Mobile nodes maintain a local copy of the most recent state of all objects, observed within a distinct area. The copies of state information on mobile nodes form a replicated database. The replicated database maintains weak consistency where mobile nodes may keep and use stale information, but any update made to a local copy will add more recent information to the database. The size of such a database is limited due to the locality of information and the resource restrictions of the mobile devices.

The synchronized clocks of observer nodes are necessary to be able to compare two or more independent observations of the same object accurately. The high accuracy of, for example, GPS clocks of approximately 360ns [GPS] is sufficient to distinguish many observations made in the real world, e.g. people's movements. It would, for example, not be accurate enough to observe the direction of a light beam passing two independent observer nodes equipped with a light sensor. In general the accuracy needed is driven by the type of observations that need to be made.

Mobile nodes use local real-time-clocks (RTC) to determine when the TTL of an observation record expires. Those clocks do not have to be synchronized, since they are only used to measure how long a record has been kept locally. Assuming a typical clock skew of a simple hardware RTC of 5 to 15 seconds per day [DALLAS], it would be sufficient to synchronize a few times a day (e.g. when passing any observer node) in order to correct the clock drift and to measure the time a record has been kept accurately enough.

All nodes are equipped with a symmetrical short range RF communication technology that offers a device discovery mechanism and allows two way communication. The RF technology is used to locally broadcast messages, i.e. every neighbor in the transmission range of the sender may receive the message. Additionally, we assume that the MAC protocol follows a CSMA/CA approach that detects collisions. Mobile nodes and observer nodes thereby form a MANET which is assumed to be partitioned very frequently due to short transmission ranges and the mobility of nodes.

3. Forwarding Strategies

For the task of distributing observations to mobile nodes a robust forwarding mechanism is needed that can cope with the frequent topology changes and network partitions in a MANET. The evaluation of flooding in such environments [HOT+99] has shown that it provides a good basis for distributing information in highly dynamic and sparsely populated MANETs. Different possibilities for flooding have been proposed and shall be briefely described here since they will be used for our algorithm presented in Section 4.

Plain Flooding: The basic version of flooding is a robust way to broadcast information in a network. Every node forwards an incoming message unless it has done so before or some knowledge of the network diameter is available to add a maximum hop count to the message. Although this is very reliable, plain flooding cannot cope with network partions or very high mobility [HOT+99].

Selective Flooding and Gossiping: Selective flooding has been proposed to reduce the number of messages in comparison to the plain flooding approach. The general idea is that a node forwards a message only to a subset of its neighbors [Tan96]. Gossiping is a variant of selective flooding where the message is sent to a subset of neighbors that is chosen randomly [HKB99]. This reduces the number of messages sent with the trade-off of being less robust, especially in networks with a low node density. Selective flooding is based on plain flooding and thus does not cope with network partitions.

Hyper Flooding is a modification of flooding proposed in [OT98]. It allows nodes to forward a message more than once if the set of neighbors changes within a given validity period of the message. This improves the delivery performance over plain flooding in scenarios with frequent topology changes (e.g. due to high mobility) and network partitions that are rejoined within the validity period of the message.

4. Negotiation-based Ad hoc Data Dissemination Protocol: NADD

This section describes an algorithm suitable for exchanging observation data in MANETs with frequent topology changes. First, data structures relevant to the algorithm are explained. Second, the algorithm itself is described. Crucial to the algorithm is when and which data is (re-)sent. A deeper discussion of selection strategies of data to be resent is presented.

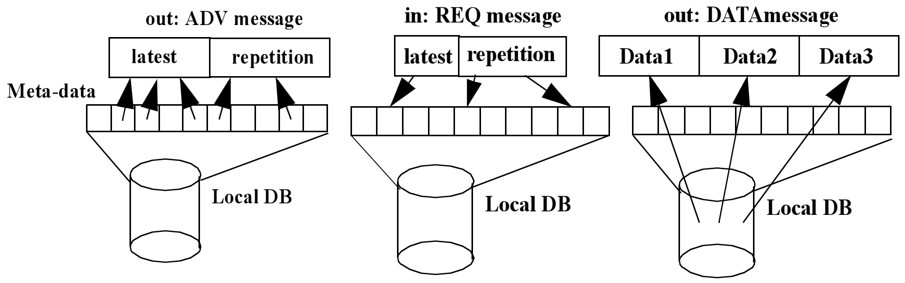

Figure 1: Interaction pattern of a node while advertising.

4.1 Data Structures

Every observer node stores an observation record for each object that is currently within its observation range. An *observation record* contains the following elements:

- Object ID (*oid*) of the observed object
- Time-to-live (*TTL*) of the observation
- Timestamp when the last observation of a state change was made (t_{obs})
- Information that indicates the replication progress of a record (*d*)
- Additional meta-data (*info*)
- State of the observed object

The *oid* kept in the observation record is a unique identifier for a real-world object such as a room or a person. Additional meta-data may be added to describe in more detail what kind of information is represented in the record, e.g. the temperature in a room or a person's position. In the context of this paper, different *oid*s represent information about distinguishable objects. The type of information represented is of no further concern for this paper. The TTL is initialized by the observer node and is continously decremented by each node that holds a copy of the record. Its initial value depends on the type of observation made (e.g. part of the meta-data) and is supplied by the observer node. The observation time t_{obs} is recorded by the observer node that has created the observation record originally, i.e. that has actually made the observation. In case of multiple observers of the same object nodes create different records about the same object. These records can be ordered due to the assumption that all observers have synchronized clocks. The precise description of *d* is given in Section 4.2.2.

4.2 Protocol

In the proposed protocol, messages are sent from a sender to all direct neighbor nodes (local broadcast). The mechanism used to forward observations is implemented using a three-way-handshake where observations stored locally in a node's database (DB) are *advertised* in ADV messages, *requested* in REQ messages from nodes that do not have the advertised information in their DB, and *sent* with DATA messages by the advertising node as shown in Figure 1. Since the state information provided by observer nodes may become large, this approach has the advantage that state data is only exchanged if at least one neighbor node requests it. Additionally, the three-way-handshake allows the optimization of advertising many observation records in a single ADV message.

An ADV message contains multiple tuples *(oid,* t_{obs}*, TTL, d)* describing information available in the DB of a node. A REQ message contains multiple tuples *(oid,* t_{obs}*)* of observation records needed by a node in response to an ADV message. A DATA message is a set of observation records that have been requested by any neighbor. Figure 3 shows an overview of the protocol in pseudo-code.

4.2.1 Interaction Between Nodes

A new information entity, i.e. a new or updated observation record that was either received by a mobile node or observed by an observer node, is offered to all neighbors of such a node by sending an ADV message. Any neighbor node may send a REQ message in return to indicate that it is interested in some of the data. On receiving a REQ message, a node broadcasts the corresponding state information. The protocol as described so far uses plain flooding on top of a three-way handshake. This results in the disadvantages of not overcoming the boundaries of network partitions as mentioned in Section 3. To disseminate information across partitions an approach similar to hyperflooding is added: whenever a node discovers a new neighbor, it is allowed to *re-advertise* observations as long as the TTL has not expired. The TTL is decremented continuously by each node that holds a copy of an observation record. If the TTL equals 0, the item is removed from the DB.

The number of items that can be advertised in a single ADV message is limited to keep messages short and thus to reduce the probability of collisions on the MAC layer. On the other hand, replication performance is improved by letting a node send more than one ADV. In our algorithm nodes may ask any node that replies to their ADV message with a request to issue another ADV message. In the current implementation a node always requests another ADV message with each REQ message sent. This process stops if no items offered in an ADV message are requested or - obviously - when the two nodes leave each others communication range. This mechanism is backed up by the creation of ADV messages if the set of neighbors of a node does not change for a predefined period of time. Figure 1 gives an overview of the basic interaction scheme.

4.2.2 Advertising Strategies

For a large number of different observations the size of each DB replica will soon be large, making it impossible to advertise all observation records in a single ADV message. Therefore an advertising node has to be able to select a subset of the data from its local DB when composing an ADV message. This leads to the problem of finding an appropriate *selection strategy* that ensures a reliable overall replication process.

As a first approach we applied a strategy mix where information that has never been advertised by a node is selected to be advertised first. If this number is smaller than the number of items an ADV message can hold, the remainder of the ADV message is filled with advertisements of data that has already been sent based on a round-robin strategy in the database. This ensures that new data has priority over data that has already been offered.

In a second class of strategies, we replaced the round-robin selection with a more sophisticated *demand driven selection policy*. When a new record is created by an observer node, it is important to give priority to the propagation of this record in order to support its replication. An approximation for that property can be made locally on any node by taking into account the number of data messages including the particular record, that have already been sent by the node. A low number of such messages indicates that not many replicas have been initiated by the node and therefore priority has to be given to that record when sending advertisements. On the other hand, this indicator is not sufficient when the record has already been replicated on almost every node. In this situation a node that received a copy of an almost completely replicated record *r* late will prefer such a record over a record *r'* that has been replicated only a few times, since the number of data messages that include *r* will soon be outrun by those that contained *r'* and will hardly increase. This is due to the fact that almost no other node will request *r* and more nodes will request *r'*. To take this into account we keep the difference *d=#adv-#data* for every record *r*, where *#adv* is the number of ADV messages sent that included *r* and *#data* is the number of DATA messages. A large *d* indicates that the record has been advertised and only relatively few requests were received that lead to DATA messages. A small *d* indicates that an item has been requested regularly in response to advertisements. To approximate the global replication progress of a particular observation record, the value of *d* calculated by other nodes is taken into account on the reception of every ADV and DATA message. The node receiving such a message re-calculates its own

$$d_{new}=(alpha*d_{old})+(1-alpha)*d_{remote},$$

where d_{old} is the previous local value for the observation and d_{remote} is the value for the same record on the node that sent the message. *alpha* is a weight, with $0<alpha<1$ that defines how much remote information is taken into account.

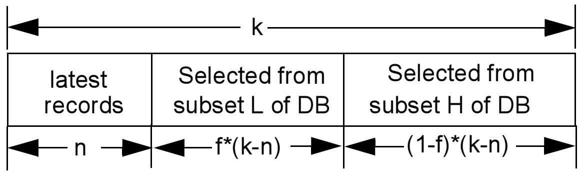

Figure 2: Structure of an ADV message for the demand driven selection strategy.

Figure 2 shows the structure of an ADV message for the second selection strategy determined by a tuple *(k, f, g)*. The message can contain at most *k* entries, where *n* are occupied by new information, just as in the round-robin selection strategy. The remainder is split into two parts, determined by the fraction f with $0<f<1$. The DB is split into two subsets *L* and *H* with

$$(DB = H \cup L) \wedge (H \cap L = \emptyset)$$

L contains a fraction *g* of all records in the local DB such that *d* for all records in *L* is smaller than the lowest *d* of any record in *H*. Records from *L* and H are selected randomly (uniform distribution) to fill f*(k-n) and (1-f)*(k-n) slots in the ADV message respectively. If any subset contains less messages, the remainder will be filled with information from the other set.

4.2.3 Randomized Messages Transmission

To reduce the number of messages and to avoid broadcast storms [NTC+99], *randomization* is used to delay messages before they are sent. ADV messages issued by mobile nodes are delayed to avoid that a group of nodes advertises the same observation at the same time and location. REQ messages are delayed, because it is sufficient that one node requests an observation, while other nodes can pick up the DATA message without requesting it. DATA messages are delayed to avoid that many nodes answer a REQ message. Delaying messages in the described way results in a flavor of selective flooding, since not every node that receives a new information entity re-advertises it. Whenever the TTL of an observation expires any node that holds it, drops it.

```
ON_NEW_DATA or ON_NEW_NEIGHBOR:
   a = prepareAdvMsg() // compose ADV message a from local DB
   schedule_for_send(a) // send within a randomized time interval

ON_RCV_ADV(m: AdvMsg):
   p = 1 // probability for ADV requesting
   r = prepareReqMsg(m, 1) // build REQ based on local DB and ADV diff
   if r contains at least one request then
      schedule_for_send(r) // send within a randomized time interval

ON_RCV_REQ(m: ReqMsg):
   d = prepareDataMsg(m) // prepare DATA based on incoming REQ
   schedule_for_send(d) // send within randomized time interval
   if m.sendAnotherAdv then // additional ADV prepared on demand
      a = prepareAdvMsg()
      schedule_for_send(a)
   fi

ON_RCV_DATA(m: DataMsg):
   store(m) // update local DB with requested data

ON_IDLE: // send messages from send buffer
   if first_item( fifo_send_queue ).send_time <= current_time
   then
      send_and_remove(first_item(fifo_send_queue))
   fi

schedule_for_send(m: msg) // send buffer with randomized schedule
   rnd = random_int(k*msg_time, 2*k*msg_time)
   if isempty(fifo_send_queue) then
      append(fifo_send_queue, {current_time+rnd, m})
   else
      append(fifo_send_queue, {last_queue_time+rnd, m})
   fi
```

Figure 3: Pseudo-code of our NADD.

5. Simulation

The proposed algorithm was tested in simulations to evaluate its performance with respect to replication latency, i.e. the time until a certain fraction of the data is replicated on all nodes, and the message overhead imposed by the algorithm.

In order to compare the discussed selection strategies, optimal selection is simulated. Nodes only advertise data that is missing in the database of other nodes. This ensures maximum efficency in the data exchange, which is only influenced by the mobility of the nodes and the underlying communication technology.

5.1 Simulation Environment

The simulations were done using a discrete time-step approach. At the MAC layer a simple carrier sense, collision avoidance mechanism (CSMA/CA) prohibits one node to send if it is within the radio range of another node that is already sending. In this case the message is scheduled to be resent later. If retransmission fails for the third time the message is dropped. If both senders are out of each others radio range, simultaneous transmissions are allowed, though the message does not reach receivers in the intersection of both ranges. If two or more senders start sending simultaneously, again messages in the intersection of any two radio ranges are extinguished and do not reach their receivers. ADV and REQ messages have a size of 32 bytes per item advertised or requested. DATA messages have a size of 512 bytes per item transfered. The transmission speed is 128 kbit/s with 10 m transmission range. Mobile nodes follow the random waypoint mobility pattern [BMJ+98] with a pedestrian speed of 3-5 km/h and intermediate stays of 72-120 seconds. Observer nodes are placed in a regular grid and remain stationary during a simulation run. The total area simulated is 100 m by 100 m to represent a large building.

In all scenarios observers can advertise at most 6 or 12 observation records per ADV message. This represents a message size of 224 (=32+6*32) or 416 bytes for ADV and REQ messages and a maximum of 3104 (=32+6*512) or 6176 bytes for DATA messages, respectively. Thereby messages for advertisements and requests are short to keep the propability of collisions low. The TTL of all observation

Scenario	Strategy	Max. ADV size	Remark
RR-6	round-robin	6	
RR-12	round-robin	12	
SEL-6	selection	6	*alpha=0.5, f=0.65, g=0.5, k=6*
SEL-12	selection	12	*alpha=0.5, f=0.65, g=0.5, k=12*
OPT-6	optimal knowledge	6	
OPT-12	optimal knowledge	12	

Table 1: Scenario overview.

records is set to a value that does not invalidate the item during the time of the simulation. All updates were done by the observer nodes at the start of the simulation. Future investigation will have to evaluate the effect of temporally overlapping replication processes.

All scenarios contain 10 mobile nodes and 9 observer nodes. Each observer node makes 80 observations, resulting in a database size of 720 observation records. The scenarios vary in the selection strategy chosen for advertisements and the maximum number of entries in an ADV message. All simulations were run for 3600 seconds. Table 1 gives an overview of the scenarios evaluated.

5.2 Replication Latency

This section presents the growth of the database copies carried on mobile nodes over time. The results of Figure 4 show the replication latency for the scenarios where at most 6 items can be advertised in an ADV message. With the optimal strategy OPT-6 it takes approximately 800 seconds to perform a complete replication on all nodes. This result solely influenced by the mobility of the mobile nodes, since each advertising node is assumed to know the contents of the database of the node it is offering data to. The round-robin strategy RR-6 uses a simple advertising schedule that only depends on what has locally been advertised before. This results in a very slow propagation, because the advertising behavior of other nodes is not taken into account at all. The demand driven strategy SEL-6 shows improvements over the round-robin strategy and results in a faster data replication, especially in the time span where 40% to 80% of the data is replicated.

Compared to the results described above, the scenario SEL-12, which uses 12 entries per ADV message shows a significantly faster growth of the database copies in the time span where 60% to 95% of the data has been replicated on each node. The optimal scenario OPT-12 shows the same behavior as its counterpart OPT-6, because it is also only limited by the mobility of the nodes. The round-robin strategy shows about the same replication latency in both cases, but varies in the message overhead as described in the next

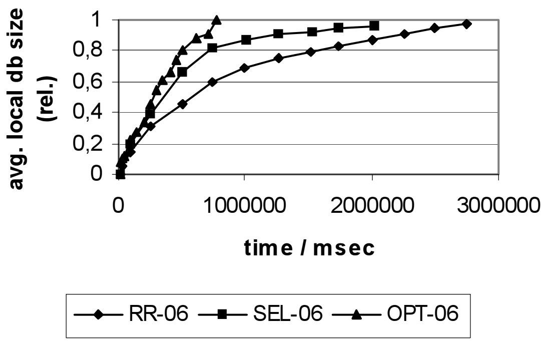

Figure 4: Average replication latency with ADV size 6.

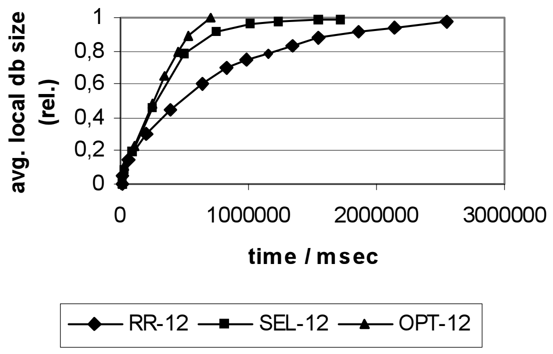

Figure 5: Average replication latency with ADV size 12.

5.3 Message Overhead

This section discusses the message overhead imposed by our protocol. The results of Table 2 give an overview of the average number of messages and their total size per node sent by each node in the different scenarios. The message sizes show the average transfer volume per node divided into ADV, REQ, and DATA volume sent. Here the messages sent have been weighted according to their size, where one ADV or REQ entry has 32 bytes, and one DATA item has 512 bytes. Each message has a constant overhead of 32 bytes. It has been assumed that every message includes the maximum of 6 or 12 entries. The optimal strategy has the lowest message overhead, since it only advertises data if necessary. It does not show the same results for ADV, REQ, and DATA messages since messages can be lost due to collision on the MAC layer and the mobility of nodes. The round-robin strategy needs about twice as many ADV messages compared to OPT. Many of those messages do not contain data that is needed and therefore only little more REQ messages are sent compared to OPT. The SEL strategy has the highest message overhead because many ADV messages contain information that is requested and therefore additional ADV messages are triggered. On the other hand this strategy shows a very good replication latency, as stated above.

Table 3 shows how much transfer volume would have been needed if, instead of the three-way-handshake, only DATA messages would have been used to propagate the observation records (i.e. in a hyper-flooding approach without negotiation). The advantage of the three-way-handshake over the plain data message approach with respect to transfer volume is 30-40% (see Table 2 and Table 3).

	Num ADV	**Num REQ**	**Num DATA**	**Size ADV**	**Size REQ**	**Size DATA**	**Total size**
RR-6	225	83	88	50kB	18kB	269kB	337kB
SEL-6	497	241	247	110kB	54kB	768kB	932kB
OPT-6	101	70	75	22kB	15kB	233kB	270kB
RR-12	168	47	50	70kB	19kB	340kB	429kB
SEL-12	377	181	176	157kB	75kB	1091kB	1323kB
OPT-12	54	37	38	22kB	15kB	237kB	274kB

Table 2: Message overhead in number and size of messages.

Assumed transfer volume using DATA instead of ADV messages	**Num DATA**	**Vol/ kB**
RR-6	225	682
SEL-6	497	1506
OPT-6	101	306
RR-12	168	1013
SEL-12	377	2273
OPT-12	54	325

Table 3: Transfer volume with DATA messages only.

6. Related Work

The SPIN protocol family [HKB99] uses a a three-way-handshake protocol similar to our protocol. SPIN addresses sensor networks, i.e. ad hoc networks with stationary nodes. Since it does not take temporary network partitions into account and therefore does not deal with the resulting problem of choosing a selection strategy the advertisement of data will only work in environments with low node mobility.

In various situations it has been proved that flooding is a robust mechanism to distribute information to all nodes in MANETs. In [HOT+99] flooding has been evaluated as a basis for multicast protocols in MANETs. Hyper-flooding has been proposed as a method to overcome network partitions in ad hoc multicast routing if, besides other parameters, the TTL for a message and the approximate network diameter are known [OT98]. In our protocol, the replicated model data already has a TTL included in its meta-data. The diameter of the network is not needed because nodes can decide to drop information solely based on the TTL, since model data is assumed to be valid for a long period of time in comparison to messages used in routing protocols. Additionally, our protocol does not perform hyper-flooding on a per-message basis but on the basis of a three-way-handshake, where advertisements are hyper-flooded by re-advertisements according to the selection strategies.

In [VB00] an epidemic protocol was introduced to solve the routing problem in a partially connected network. They use a similar mechanism to exchange information between two neighbor nodes. However, their goal is to deliver messages to any node without establishing a route between sender and receiver and not the replication of model data. The data considered is typically short-lived, i.e. if routing of a message fails a retransmission is issued.

A combination of so called rumor-mongering and anti-entropy is used in [DGH+87] to replicate information in databases in wired networks. In our protocol, we combine new information and, if free space is available in an ADV messages, older information. This results in a partial anti-entropy session, because some differences between hosts are resolved with new information first (i.e. rumors) and older information (i.e. part of the anti-entropy).

In [XWC00] the distance between any two versions of a data item and the communication cost is used as the basis for a cost model in order to determine the estimated benefit of forwarding the data. In this approach it is necessary that every node has a notion of „distance“ which depends on the semantics of the data. The authors also make the assumtion that only a single node updates a particular object.

7. Conclusion

We presented a protocol for information dissemination in mobile ad hoc networks. The protocol replicates information stored in local databases of nodes. In order to reduce the data transfer volume, negotiation is used to advertise and request data among mobile nodes. Network partitions, as they appear due to node mobility or low node density, can be tolerated since data is advertised more than once. The selection strategy that determines which data is re-advertised, influences the performance of the protocol with respect to the propagation latency and the data transfer volume. The demand driven selection policy shows a reduction of the data transfer volume by 30 to 40% compared to a plain hyper-flooding approach which does not use negotiation. The replication latency performs nearly optimal till 80% replication of the data is achieved and slows down for the last 20%.

So far, we have investigated the impact of different data selection strategies on replication latency and message overhead. In future work, we will investigate what parameters can be used to adjust the hyper-flooding nature of advertisements, e.g. depending on the node density, in order to achieve further reduction of advertisement messages in dense networks. Mobility models of mobile nodes have impact on the performance of routing protocols [THB+02]. We will examine the impact of mobility models on our protocol and the improvements that can be made if such knowledge is exploited.

References

[BMJ+98] Broch, J., Maltz, D. A., Johnson, D. B., Hu, Y., Jetcheva, J.: „A Performance Comparison of Multi-Hop Wireless Ad Hoc Network Routing Protocols“, Proceedings of the Fourth Annual ACM/IEEE International Conference on Mobile Computing and Networking (MobiCom’98), 1998.

[DALLAS] Accuracy of RTC: http://dbserv.maxim-ic.com/appnotes.cfm/appnote_number/632

[DGH+87] Demers, A., Greene, D., Hauser, C., Irish, W., Larson, J., Shenker, S., Sturgis, H., Swinehart, D., Terry, D.; „Epidemic Algorithms for Replicated Database Maintainance“, Proceedings of the 6th ACM Symposium on Principles of Distributed Computing, 1987.

[GPS] Information GPS: http://www.colorado.edu/geography/gcraft/notes/gps/gps_f.html

[HKB99] Heinzelman, W. R., Kulik, J., Balakrishnan, H.: „Adaptive Protocols for Information Dissemination in Wireless Sensor Networks“, Proceedings of MobiCom, 1999.

[HOT+99] Ho, C., Obraczka, K., Tsudik, G., Viswanath, K.: „Flooding for Reliable Multicast in Multi-Hop Ad Hoc Networks“, Proceedings of MobiCom Workshop on Discrete Algorithms and Methods for Mobility (DialM’99), 1999.

[KMP99] Karumanchi, G., Muralidharan, S., Prakash, R.:“Information Dissemination in Partitionable Mobile Ad Hoc Networks“, Proceedings of 18th IEEE Symposium on Reliable Distributed Systems, 1999.

[NTC+99] Ni, S.-Y, Tseng, Y.-C, Chen, Y.-S., Sheu, J.-P.: “The Broadcast Storm Problem in a Mobile Ad Hoc Network“, Proceedings of the Fifth Annual International Conference on Mobile Computing and Networking (MobiCom'99), 1999.

[OT98] Obraczka, K., Tsudik, G.: „Multicast Routing Issues in Ad Hoc Networks“, Proceedings of the IEEE International Conference on Universal Personal Communication (ICUPC’98), 1998.

[Roem01] Römer, K.: „Time Synchronization in Ad Hoc Networks“, Proceedings of the ACM Symposium on Mobile Ad Hoc Networking and Computing (MobiHoc’01), 2001.

[Tan96] Tanenbaum, A.: „Computer Networks - Third Edition“, Prentice Hall, 1996.

[THB+02] Tian, J., Hähner, J., Becker, C., Stepanov, I., Rothermel, K.: „Graph-Based Mobility Model for Mobile Ad Hoc Network Simulation“, Proceedings of the 35th Annual Simulation Symposium, 2002.

[VB00] Vahdat, A., Becker, D.: „Epidemic Routing for Partially-Connected Ad Hoc Networks“, Technical Report CS-200006, Duke University, 2000.

[XWC00] Xu, B., Wolfson, O., Chamberlain, S.: „Spatially Distributed Databases on Sensors“, Proceedings of the 8th ACM Symposium on Advances in Geographic Information Systems (ACMGIS’00), 2000.