Utilizing a fault-tolerance protocol for colocating interfering cells in a wireless network

Spiro Trikaliotis, Jörg Diederich
spiro@ivs.cs.uni-magdeburg.de, joerg.diederich@graffiti.net
Institute for Distributed Systems
University of Magdeburg, Germany

Abstract: Achieving quality-of-service (QoS) in multi-hop wireless networks is still an open research topic. Our approach uses clusters which are located directly nearby, resulting in overlapping transmission areas. Such overlapping transmission areas are prone to the so-called "hidden station problem". This paper shows the results of an experiment which shows that this effect can be highly reduced with the help of a fault-tolerance protocol, namely, our "reliable group-communication protocol" (RGCP).

1 Introduction

In disaster situations, it is crucial for the aid to be able to be as fast and coordinated as possible. For this purpose, communcation between all aiders has to be fast and reliable. Unfortunately, one cannot depend on any infrastructure, as the disaster might have destroyed that, too. Voice-over-IP (VoIP) is a possible solution for this, if IP communication can be set up without pre-built infrastructure. Mobile Ad-Hoc Networks (MANET) allow to establish such a communication without the need for an infrastructure. The network organizes itself to be able to deliver information over many hops of wireles stations, without help from the user or some infrastructure backbone. Now, applications like voice-over-IP (VoIP) are asking for quality-of-service (QoS) in the underlying network in order to operate correctly. Currently, mostly infrastructure networks are used, where at most the last connecting link is done on a wireless medium. This approach has the advantage that QoS can be achieved with already known mechanisms which are used for wired network, with only small or even no additions for the wireless part of the connection. QoS for MANETs is a relatively new area of research. In fact, most current MANETs do not allow for QoS guarantees, as achieving quality of service in a multi-hop ad-hoc network is a non-trivial task. Obviously, VoIP asks for QoS: VoIP needs some guaranteed bandwidth and some maximum delay in order to be usable.

As the network topology is potentially changing at a rapid pace, achieving QoS in such a scenario cannot be achieved without some properties of the underlying routing protocol. For example, this routing protocol has to be very fault-tolerant in order to tolerate the high loss rate of network traffic which is to be expected in this scenario.

Our approach [Tri04] to achieve QoS in a MANET is based on the fact that the problems

in achieving QoS in ad-hoc networks mostly arise because of the potentially high number of involved and moving nodes, which makes planning very hard for the protocol. Furthermore, routing protocols for ad-hoc networking tend to use reactive approaches. With these approaches, information on the network structure is only sampled when it is necessary. This reduces the network utilization and energy consumption of the nodes significantly, which is very important for wireless scenarios. Unfortunately, reactive approaches do not carry enough information to be able to react to moving stations in a manner that the given QoS guarantees are not violated. Our approach partitions the network into "natural" clusters of nodes which consist of nodes which are very likely to remain nearby for the future. This allows the network to gain a rather stable structure which helps in reducing the network traffic needed to hold the network together.

Due to the limited reachability of the wireless medium, not every station can talk with each other. Because of this, the communication in wireless networks is often looked at as communication of cells, that is, of stations which are in direct reach of each other. The clusters mentioned above are such cells. When two such cells are colocated, that is, located one directly by the other, there are stations of both cells which can talk to each other. Additionally, there are stations which cannot see the stations from the other cell. This imposes a problem, the so-called "hidden station problem", because the stations cannot determine if their messages will collide with each other.

Our approach distinguishes between communication inside of a cluster (intra-cluster) and between different clusters (inter-cluster). The intra-cluster communication is based on a proactive protocol. Each station inside of the cluster is polled one after the other, and the information is broadcast in the cluster. This way, every station has equal knowledge about what data is available in the cluster. This intra-cluster communication is done with a variant of RGCP [NS03], a protocol which is designed for just that type of communication. Additionally, RGCP is designed with fault-tolerance in mind. This means, it can tolerate many losses without affecting the delivery of messages. With many colocated cells which do not control each other, this is very important.

The inter-cluster communication is done via a reactive protocol, thus, data is only collected when it is needed. This protocol is out of the scope of this paper. Later papers will deal with this one.

Thus, in this paper, we show a setup to measure the throughput of RGCP in the case of colocated cells. Beside the throughput, the application losses and media losses are monitored, too, to be able to appreciate the measured values of throughput.

This paper is organized as follows: Section 2 presents a short description of the RGCP protocol, while section 3 describes the used implementation. In section 4, the experiment setup is presented. The results are shown in section 5. Finally, section 6 closes this paper.

2 The protocol RGCP

RGCP is a protocol for reliable group communication. That is, every station is aware of the group it is communicating with, and the protocol ensures that all correct stations deliver

the same set of messages in the same order.

To achieve this, this protocol uses a round-based approach. A central station, the coordinator, polls every station in its group, granting it the token which allows the station to send. The station getting the token answers with data to be sent to the group. The coordinator takes this information and broadcasts it to the whole group. After this, the coordinator proceeds with the next station, until it has polled every station in the group. Subsequently, the next round is started, polling the first station.

Thus, to summarize, in RGCP, a round consists of the following actions:

- For each station A in the group:
  1. The coordinator polls station A for any data (POLL)
  2. Station A answers to the coordinator with the data it has to sent (Request Unit, RQU)
  3. The coordinator broadcasts this data to all stations in the group (broadcast unit, BCU)

This implements the main channel in RGCP communication. As RGCP has to ensure atomicity, ordering of all information, fault-tolerance and the group membership, there is more data to be sent. This is done via sub-channels, data which is sent piggy-back on all three types of data (POLL, RQU, BCU). [NS03], [MNS99] explain this in more detail.

RGCP is designed to maintain fault-tolerance. That is, it handles message losses by introducing redundancy whenever needed. For this purpose, a scheme is used which utilizes sequence numbers as well as explicit acknowledges sent from the group members to the coordinator (in a RQU) and decisions sent from the coordinator to the groups members (via a BCU). The details are beyond the scope of this paper; they are explained in [MNS99].

If no message losses occur, this protocol does not impose much overhead, only some bits which are sent piggy-back are used additionally to the data to be sent. The central coordinator allows for a stable reachability of all nodes.

Whenever message losses occur, the station has to re-send its RQU or the coordinator has to re-send its BCU, depending upon which message got lost. This adds some overhead to the protocol to ensure the delivery of data. If RGCP is used stand-alone, this overhead is not very big, as message losses do not occur that frequently in this scenario. The central coordinator makes sure every station gets the token to send one after another, serializing the access of the stations to the medium and significantly reducing the probability of collisions.

Anyway, in a scenario with many colocated coordinators, this is not true anymore. In every cell, there is one station holding the token to send, thus, the send data frames compete for the wireless medium, resulting in media losses. The fault-tolerance parts of RGCP have to make sure no data is lost at the application level. To achieve this, data is re-sent on a quite regular basis, resulting in lower bandwidth.

The important question is: How much does this affect the available bandwidth? Does the bandwidth deteriorate much more than the physical limits dictate? With n colocated cells,

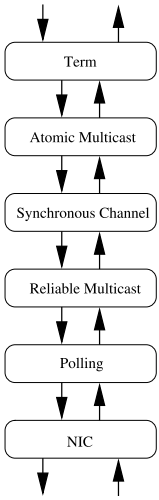

Figure 1: Microprotocol stack

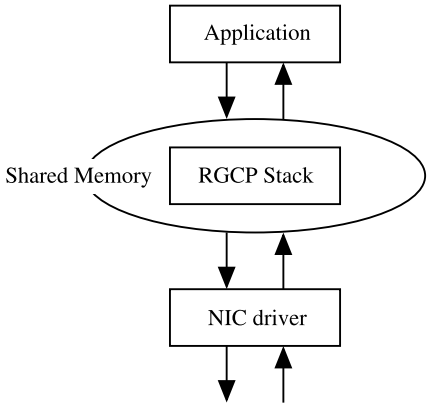

Figure 2: Overall architecture

we expect the bandwidth per cell to fall to $1/n$ of all available bandwidth, as the sum of the bandwidth of all cells cannot be bigger than the physical bandwidth available. Anyhow, a good algorithm should not be much worse than that.

3 Implementation architecture

RGCP implements some features (like atomicity, order, and fault-tolerance) which do not necessarily depend upon each other. Thus, it seems logical to divide it into several smaller parts which can be designed, tested and used separately. RGCP is built with such a microprotocol architecture in mind. The RGCP implementation consists of several modules, the microprotocols, as described in [Van04]. Each one provides a communication feature of RGCP. Like in the ISO/OSI model, the microprotocols are arranged in layers which are implemented in a protocol stack. Protocol functions not needed may easily be disabled by taking out the appropriate microprotocols of the stack (unlike in the OSI model).

Figure 1 shows the arrangement of microprotocols used for RGCP and thus, in this experiment. The uppermost protocol provides the interface to applications, while the lowest provides the interface to the network driver. For each microprotocol two different implementations exist, as a station can perform one of two different tasks in a running network, a coordinator or an ordinary station. These different implementations can not be mixed inside a stack. However, the role can change while operating.

Looking at the OSI-model, the protocol stack can be considered as a dispatcher between components working at upper and lower layers. The shared memory is intended to be used

by these components, too. An application, which uses RGCP to communicate with other applications, communicates with the protocol stack via the shared memory. At the bottom, the network device (NIC) driver is attached in the same way.

The complete architecture of the implementation is shown in figure 2.

4 Measured values

Depending on the kind of data they are related to, the determination of measured values is done in very different ways. As throughput and application losses are related to application data, both are measured with the help of an appropriate application. Media losses detecte by RGCP may be determined best at the detection mechanisms itself inside of the protocol stack

4.1 Throughput

Like in many benchmarks and tests, the main value of interest is the maximum throughput. Thus, the task was to transmit as much data as possible from one application to other applications, within a certain amount of time.

A transmission requires one sender and - at least - one receiver. Both were implemented to run independently from each other in the measuring application. The same application was running on all clients during the experiment. As shown in section 3, the application exchanges application data with the protocol stack via the shared memory. With respect to this additional component, sender and receiver had clear tasks:

- At any time, the sender had to ensure, that the shared memory contains at least one unprocessed data packet in the outgoing queue.
- The receiver raised a simple counter if new application data were available. It was informed of this case by the shared memory implementation.

It showed that new data packets were filled much quicker into the shared memory than they were processed and taken out by the protocol stack. In order to avoid continous overloads, sender and shared memory were synchronized. If the outgoing queue of the shared memory was completly filled, the sender stopped it's work until 40% of the queue were freed again.

The described proceeding would only enable a throughput value in number of packets per time, which is a highly dependent value. For simplicity, all packets received (and sent out, of course) contained the same payload length. With this information, the desired throughput value was computable in bytes per second.

4.2 Application losses

Although RGCP provides reliability, sent application data still can get lost. This is possible in case the assumptions on the used radio network, which in turn are used to configure the protocol stack, do not fit with reality.

If atomarity is enabled, application losses get a special meaning. Atomarity ensures either all applications on all clients receive certain data, or none does. Therefore, an application loss signalizes that at least one client did not acknowledge the receipt of data. In fact, even if all but one client were successful, this results in an application loss.

The fixed order of data transmissions in conjunction with receiving the own data allows each station to detect application losses. As an application requests RGCP to transmit data to the whole group, it may receive its own data back in return. This is the normal behaviour of RGCP: data is sent from a client to the access point, which broadcasts this data to the whole group. Each cooperation between access point and client treats data in a strict sequential order. Only if the processing of one unit is finished, the next will be processed. Knowing this, an application could sign outgoing data with a strict increasing sequence number in order to detect losses of own sent data. Whenever the difference of sequence numbers between received data is more than one, at least one loss has occured. In fact, the difference minus one tells the number of successive losses.

This implementation was done with the application mentioned before in section 4.1. Knowing the first sequence number ever sent out even allows it to detect the loss of the data packets first sent.

4.3 Media losses

As mentioned in section 2, RGCP uses a round-based communication approach. Each client communicates with the appropriate access point exactly once in a round. This communication is divided into three parts: poll, request and broadcast. The first and the last one, that means poll and broadcast, are done by the access point with one transmission each. The request is done by a client with a transmission, too. Of course, all three transmissions could fail due to errors on the media.

The detection of losses uses the error detection mechanisms of RGCP. The possibilities to detect losses are as follows:

- Poll
  Each poll transmission contains the actual round number. This number is sequentially increasing. So, a client could easily detect losses by comparing the round numbers of the actual received poll and of the poll received last. The difference – minus one – tells the number of successive losses.

- Request unit
  From a certain point of view, the request transmission is an acknowledgment of a

client to a previous poll message. Because the poll message could get lost before, the absence of a request does not necessary mean it was lost.Therefore the loss detection at the access point is not possible.

The loss detection at the originating client is not possible too. A request is indeed followed by a broadcast sent by the access point. But this broadcast is no direct reaction to a received request, therefore it is not useable as any kind of acknowledgment. Additionally, this broadcast could get lost as well.

- Broadcast
  Just like the poll messages, the access point marks all broadcasts with an increasing sequence number. So the proceeding in order to detect a loss of one or more broadcasts is the same as it is for poll messages.

The measurement of broadcast losses can exactly be done by each client, as explained before. The problem in determining request losses can be solved by treating poll and request messages as one unit. Since a request is a reaction to a poll, it can only get lost as a poll was received. In reverse, if a poll got lost, there's no request which can get lost. This means, if a loss of one of these both messages occurs, it is exactly a loss of one of them. Unfortunatly it can't be said - by the sender of the poll - which message got lost, whether poll or request.

At least in theory, it would be possible to distinguish between lost poll and request messages. For this, the clients had to measure lost polls, as described in the first point above, and notice the missing round numbers. The appropriate access point has to do the same for each poll/request loss. The intersection of the collected numbers by each client and the access point would present lost polls, and the remaining numbers represent lost requests.

In detail, the measurement of poll/request losses happens at each access point. Here poll messages are sent out and request messages are received. After sending a poll, an access point expects the receiving of a request within a certain time. According to the thoughts just presented, if the time elapsed without a receipt, exactly one media loss has occured.

5 Results

The main idea of the experiment was to determine the influence of multiple active groups on each other. An active group in terms of the experiment consists of one access point station and one or more client stations. All stations were creating network traffic during the whole time of measurement.

Due to design intention and implementation, as shown in sections 2 and 3, RGCP is very flexible and adjustable. Together with different configuration possibilities of each group, the number of global configurations increases dramatically. The following restrictions were specified in order to determine possible configurations:

- every group consists of one client station and the respective access point station

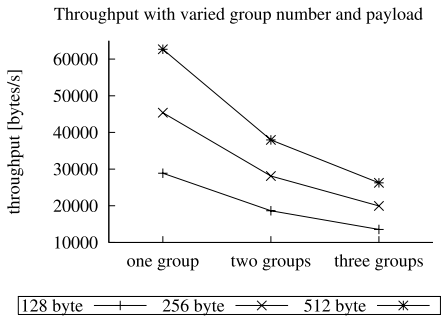

Figure 3: Development of throughput

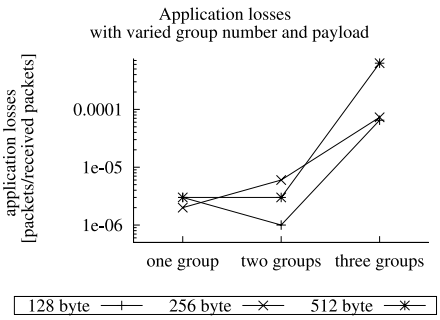

Figure 4: Development of application losses

- every group transmits data with the same payload
- every station of every group is configured with the same settings for tolerated successive media losses, the so-called omission degree (OD)
- every group runs for the same duration as other groups. All groups start simultaneously and end simultaneously.
- all groups communicate at the same channel and in direct reach of each other. Because of this, every station can send to and receive from every other station.

This experiment was run with one, two and three simultaneous active groups. For each of this three possibilities, the payload of transmitted data was varied from 128 to 256 to 512 bytes. Taken this all together, nine different configurations were used in this experiment. Each run took 10 minutes. In order to receive a representative set of data, the runs of every configuration were done 5 times.

Additionally, the composition of groups was varied in order to reduce possible influences by the used hardware. This was done by selecting different computer systems for the respective groups. So, one computer could have been client or access point in different groups in the same configuration.

The results of all five runs done with a certain group composition were averaged. Finally, the averaged results of different group compositions within a configuration were averaged, too.

Figure 3 shows the development of data throughput as the number of active groups increases.

With every additional group, the throughput significantly decreases. In comparision to a single active group, the throughput of each of two active groups decreases by more than 35 percent. In comparison of two and three active groups, the throughput decreases by more than 27 percent. Using a higher payload slightly enforces the decrease by approximatly 1.8 percent.

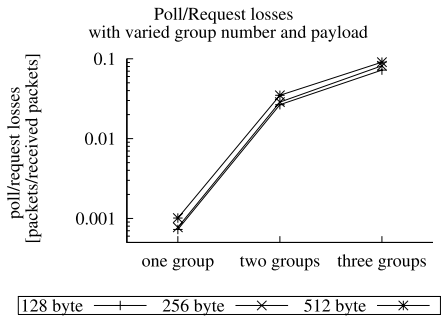

Figure 5: Development of poll/request losses

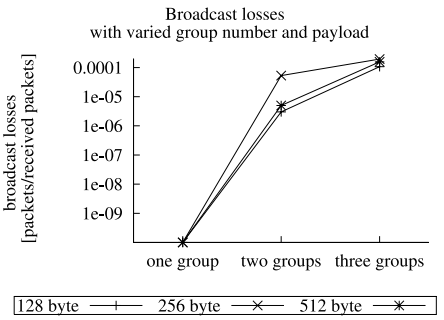

Figure 6: Development of broadcast losses

In order to be comparable to the measurments done in [Sch00], loss results were related to the number of received packets. This proceeding allows an estimation about the necessary number of packets for a loss to occur. The more intuitive approach to relate losses to the number of transmitted packets wasn't realizeable. Because of the buffer structure explained in 3, only the number of received packets could be determined for shure. The development of application losses is shown in figure 4.

As mentioned in 4.3, two kinds of media losses were measured. The development of poll/request losses shows figure 5, and figure 6 does the same for broadcast losses.

For a better visual understanding, a logarithmic scale of the loss results was choosen. Not surprising, all figures show dramatically increases of all loss values, as more groups are active. In comarision between both kinds of media losses, poll/request losses happen a lot more often than broadcast losses. Application losses approximatly happen in the same regions as broadcast losses.

6 Conclusion

This paper showed the behaviour of a cell-based fault-tolerant protocol for sending data to the cell members in a "foreign" scenario, that is, in a scenario where many of these cells are colocated. This protocol was never intended to be used this way. Anyway, the fault-tolerance aspects of the protocol make it behave in a well-tempered way even in this scenario.

This protocol will be used in such a scenario to provide quality of service for the communication of nodes in a mobile ad-hoc network.

References

[MNS99] M. Mock, E. Nett, and St. Schemmer. Efficient Reliable Real-Time Communication for Wireless Local Area Networks. In *Third European Dependable Computing Conference EDCC-3*, pages 380–397, September 1999.

[NS03] E. Nett and St. Schemmer. Reliable Real-Time Communication in Cooperative Mobile Applications. In *IEEE Transactions on Computers*, volume 52(2), pages 166–180, 2003.

[Sch00] Stefan Schemmer. Zuverlässige Echtzeit-Gruppenkommunikation auf einem lokalem Funknetz. Diploma thesis, Institute for Autonomous Intelligent Systems (AIS) of the GMD in cooperation with the Institute for Distributed Systems (IVS) of the Otto-von-Guericke-University Magdeburg, January 2000.

[Tri04] Spiro Trikaliotis. Utilizing Fault-Tolerance for Achieving QoS in Ad-hoc Networks. In *Workshop Proceedings Dependability and Fault-Tolerance, Organic and Pervasive Computing, International Conference on Architecture of Computing Systems (ARCS)*, pages 66–75, March 2004.

[Van04] Sebastian Vandersee. Effiziente Realisierung in SDL spezifizierter Mikroprotokoll-Architekturen. Diploma thesis, Institute for Distributed Systems (IVS), Otto-von-Guericke-University Magdeburg, April 2004.