Utilizing Fault-Tolerance for achieving QoS in Ad-hoc networks

Spiro Trikaliotis

Institute for Distributed Systems (IVS)
University of Magdeburg
Universitätsplatz 2
39106 Magdeburg
spiro@ivs.cs.uni-magdeburg.de

Abstract: Ad-hoc networks consist of many independent wireless nodes which communicate without the help of an infrastructure. Since this type of networks exhibits a dynamic topology, that is, the nodes move very frequently, it is hard to establish some quality-of-service (QoS) in this scenario. This paper presents an approach which accomplishes this. QoS is guaranteed in the sense that the network either can offer these parameters, or it informs the source that it cannot do this anymore. While this guarantee is not so stringent that the network really can guarantee it in every case, in most cases, a QoS degradation as well as a link break will be announced to the sender beforehand, allowing it and the application to take some counter-measures such as putting the system in a safe state before the QoS degrades.

1 Introduction

Ad-hoc networks are wireless networks without infrastructure. In general, mobile stations build a routing web which propagates information to the destination. The field of ad-hoc networking is growing constantly and getting more and more into interest.

Ad-hoc networks have many advantages over infrastructure networks, for example in catastrophic scenarios. Using current cellular phones, there is a need for an infrastructure, but that might have gone havoc by the catastrophe. Consider the case of a fire department in a forest fire. The fire fighters can wear sensors, as well as place them at strategic positions, or these might be thrown out of planes. These sensors, which can include sensors as small as temperature sensors or as big as (infrared) cameras, build a network transporting their values to every fire fighter who is interested in these. With this approach, the fighters are helped in deciding what to do, and they can be warned when their way back is at risk of being cut off.

Like any new technique, ad-hoc networks have many unresolved issues. First of all, there are issues due to the wireless nature. Other than their wired counterparts, wireless networks suffer from comparably high message loss on the wireless medium. Furthermore, ad-hoc networks add another layer of problems onto this. In wireless networks with infrastructure, like cellular phones or WLANs in infrastructure-mode, the infrastructure is wired and fixed. Because of this, routing is not such a big problem. The infrastructure is not likely to change, thus, routing algorithms can be run which are similar to those of wired networks like the internet. Mobile stations moving around and changing their point of access to the wired infrastructure pose little problems in this scenario. The usual solution is to have centralized data bases, i.e. as used by cellular phones [Sc03]. Unfortunately, this solution is not feasible in an ad-hoc scenario, as there is no central station. This paper presents an approach which achieves QoS in an ad-hoc scenario without utilizing a centralized infrastructure. This QoS cannot be enforced, as it is not possible to enforce QoS in an ad-hoc network since the topology can change randomly and unpredictable. Anyway, the network can tell if the current topology can meet the desired QoS parameters now, and the network can tell whenever there are significant changes to the topology, possibly breaking QoS now or in the future. This is the kind of guarantee our solution addresses.

The outline of this paper is as follows: Section 2 presents the prerequisites and the outline of our approach. Section 3 briefly discusses some related work, on which our approach is based partly, while a more detailed discussion of our approach is presented in section 4. Finally, section 5 concludes our paper.

2 Approach

2.1 Prerequisites

Whenever we speak of QoS, we have to define which type of QoS is meant. In the following, we are interested in the following type of QoS: After the sender successfully queried the network whether the network can guarantee this, that is, the network responded positively, the following holds true:

1. *Maximum transmission time*. If a sender p has been guaranteed a maximum transmission time of Δ, whenever it sends a message at an instant of time t_{send}, this message arrives at the receiver at an instant of time $t_{arrival}$ with $t_{send} \leq t_{arrival} \leq t_{send} + \Delta$, or p has to be informed that this will not be possible anymore in the future.

2. *Bandwidth*. If a sender p has been guaranteed a bandwidth of B per time t, the network ensures that this bandwidth can be send, or the sender p has to be informed that this guarantee cannot be held in the future.

3. *Reliability*. Unless a network partition occurs, information put on the network by the sender should be received at the destination. If this is not possible, the sender has to be informed.

Informing the sender in case the parameters cannot be guaranteed anymore should be done as soon as possible. In most cases, this means it should be done even before the violation occurs. Since the network behaves in a non-predictable way, the network cannot foresee if it won't be able to satisfy the QoS in the future.

The QoS should only be applicable once a connection is established; the establishment of a connection does not need to behave in a QoS way. This restriction is fine in most scenarios. Furthermore, we are already used to it – think of calling your co-workers mobile: It can be delayed for some time.

There is one assumption throughout this protocol which models the behavior of many real ad-hoc networks:

- *Bounded omission property*. There exists a so-called *omission degree OD* such that: If a sender sends the same message *OD*+1 times, this message will arrive at the receiver, or the link has broken.

Furthermore, for better control of the network, we assume that the underlying network does not resend failed messages if they did not arrive or arrived with invalid error checks. This assumption can be made true even on WLANs based on 802.11, which normally perform resending of messages, if every information is send as a broadcast.

2.2 Outline of our approach

Most existing network routing protocols for wired networks are so-called proactive approaches, that is, all stations exchange information about the whole network. This enables every station to know how to route to every other station at every instant point of time. While this is feasible in wired networks which are almost static by nature, this is not the case with ad-hoc networks where stations change locations very frequently. Even if there's nothing to be transferred, every change in the network topology would have to be propagated. Due to the high dynamic in ad-hoc networks in conjunction with the low available bandwidth wireless network provide compared to wired networks, these proactive protocols eat up almost all bandwidth and do not allow for much data transmission at all. Because of this, most existing ad-hoc networks use a reactive approach, that is, routes are only looked for when they are currently needed. If a route breaks, a new route is searched again [Pe01].

While the reactive approach ensures not to collect too much information which might become stale very soon, it is not suitable for QoS routing since generally, these approaches rely on one single route. A single link break along this route requires repairing that route, most often implemented by telling the sender to search for another route or something similar. This would break any QoS guarantee the network might have given to the communicating pair.

Our approach for QoS routing in ad-hoc networks utilizes a hybrid approach. We build clusters of stations, that is, colocated stations are treated as one cluster. We use a proactive group communication protocol as routing protocol inside of the cluster (intra-cluster). This results in cluster-global knowledge, that is, every station inside of the cluster has the same knowledge. This allows using local fault tolerance inside of the cluster, and allows for treating the cluster as a kind of a "super-node", reducing the amount of information needed for the network topology. This intra-cluster protocol is designed in a way such that the time used for communication is preplanned. This way, the cluster knows how much time and bandwidth there is to be able to fulfill QoS needs. The information that stations changed location from one cluster to another doesn't need to be propagated all over the network, but it can be handled locally.

On top of the intra-cluster routing, the inter-cluster routing connects the different clusters with each other. The inter-cluster routing protocol is reactive, clusters only exchange information about their locations when there is information to be spread. When this happens, the cluster containing the source searches paths to the cluster containing the destination. Furthermore, it looks out for alternative paths being able to take over in case the primary path breaks. The paths are selected and stored based on clusters, not based on stations. This way, inter-cluster routing does not need to know about changes inside of the clusters as long as the clusters can be reached.

The inter-cluster routing algorithm is not entirely reactive. It needs proactively monitoring of the primary and backup paths in order to notice when they are about to break, or when they are broken. A breaking primary path has to be replaced with a backup route, a breaking backup route has to be reestablished as soon as possible. Of course, the extra time for switching to backup routes has to be taken into account from the beginning, allowing not to violate QoS parameters in this case.

The network gives the sender a QoS guarantee based on the QoS information inside of every cluster through which it wants to route, the number of clusters which have to be crossed, taking in consideration the possibly longer paths of the alternative routes, and a safety guard space. This guard is needed since stations moving around might make the path longer than it had been before. When the network encounters such a case, it informs the source that the QoS guarantee might be violated in the future, giving another, less stringent guarantee, so the communication has the opportunity to adjust itself to this new guarantee.

The routing protocol has to be divided in the following subproblems which have to be solved:

1. Dividing the network into clusters;

2. Routing inside of a cluster (intra-cluster routing);

3. Communication of a cluster with his neighbored clusters;

4. Synchronization of adjacent clusters such that their (pre-planned) communication time does not overlap;

5. Routing based on the clusters (inter-cluster routing);

6. Management for station movements, route changes and route breakage.

Due to space limitations, this paper cannot present each of these aspects in detail. This is the reason why we will focus on the inter-cluster routing in this paper, and will make references on other papers where appropriate. Some parts have not been presented to date, so they will the topic of subsequent papers.

3 Related work

While there is much research regarding routing in ad-hoc networks, there is little work known to us regarding QoS other than using priority-based schemes. [Pe01] and [LK03] contain some state-of-the-art algorithms on routing in ad-hoc networks, namely dynamic source routing (DSR [JMB01]), destination sequenced distance vector routing (DSDV, [PB94]), ad-hoc on-demand distance vector routing (AODV [PBD03]), temporally ordered routing algorithm (TORA [PC97]), to name just a few. These protocols are all happy having found one route from the source to the destination.

There are multipath protocols which try to find more than one path to a destination. The problem in searching multipaths lies in that routing loops are to be avoided. Ad-hoc on-demand multipath distance vector (AOMDV, [MD01]) ads multipath to AODV. AODV searches for a route by flooding the network with a request. Every station seeing the request forwards it, unless that station has seen the very same request. AOMDV lessens this rule by using a hop-count from the source to every station. Whenever a station sees a request a second time, it checks if the previously seen hop-count is bigger than the new one. If this is the case, the new route is shorter and thus, cannot be part of a loop. Unfortunately, this rules out many legitimate paths since it is likely that a route with a bigger hop-count takes more time to advance to a station. Nishibayashi et al. [NSK03] propose another approach, where every node forwarding a request adds its own identification to the packet. This way, loops can be recognized just by looking at the route taken so far: If a node recognizes itself, it discards the packet as it has been gone through a loop.

The approach of clustering in ad-hoc networks is not new [St01]. Clusters are built for transmission management, building a routing backbone or for routing efficiency. Furthermore, they can be built to minimize communication costs [BC04]. Nevertheless, we are not aware of any cluster-based approach trying to achieve QoS.

4 The routing protocols

As we already have stated already in section 2.2 above, we will focus on the inter-cluster routing in this chapter. Nonetheless, a brief overview of the other parts follows.

Dividing the network into clusters is crucial, as it determines the way each and every node is handled as part of the whole protocol. Anyway, choosing the right way to cluster the network depends on the characteristics of the routing protocols. The clusterheads are chosen based on the number of their neighboring nodes to form clusters which are not too small. Then, the cluster is defined to be the clusterhead and every station which can be directly reached by the clusterhead, that is, without any other node having to forward each frame.

Each cluster has to be able to deliver the messages reliably and timely. Because of this, we chose to use the protocol RGCP [NS03]. In detail, we use the reliable group communication part of this protocol which allows us to treat every cluster as a kind of super-node due to the cluster-global knowledge inside of each cluster. The RGCP protocol has the following structure: In each cluster, the clusterhead has a special importance. It has priority on the network, and thus is able to poll each and every station. [NS03] explains the details. The *bounded omission property* (see above) and the fact that the clusterhead knows the number of nodes in his cluster ensures that every station which has a working link to the clusterhead will eventually get every information the clusterhead sends, and vice versa. These facts even allow to determine the maximum propagation delay of each cluster. It should be remarked that the network is not used in polled state for the whole time, only for part of it.

If a node leaves the cluster, the cluster cannot guarantee the timely delivery of information anymore. This case is handled by the inter-cluster routing protocol, see below. Another point to concern is a clusterhead which increases its distance from the rest of the cluster, until the stations cannot reach the clusterhead anymore. This is part of the management of station movements which will be explained below.

If information has to leave one cluster and is intended for another one, two clusters have to communicate with each other. This communication is implemented in so-called gateway nodes. These are nodes belonging to one cluster which are in reach of another cluster. They inform the clusterhead which clusters can be reached via themselves, thus giving the clusterhead a local view of its neighborhood. If the clusterhead wants to send information to another cluster, it sends it to one of his gateways, which sends the information to a gateway of the other cluster accordingly. We want to remark that a clusterhead itself might become its own gateway node.

The protocol utilizes clusters which are overlapping in many cases. This poses a problem on the intra-cluster protocol which uses polling: Since two clusterheads have overlapping reach, two adjacent clusters may disturb each other while polling, resulting in much more message losses. Our approach takes this into account in two ways: First, the omission degree has to be high enough to tolerate such losses. Second, our approach synchronizes adjacent clusterheads such that they are not polling their stations at the same instant of time. This is accomplished with the help of the gateway nodes. Every gateway node tells the clusterhead which part of the time is used by other sending stations. This allows the clusterhead to plan its time to reduce the amount of overlapping as far as possible. Since the time of the polling can be pre-planned, this can be accomplished. Furthermore, the clusterhead tries to reserve some time for the gateway to be able to send information to other gateways.

On this ground, the inter-cluster routing algorithm has to find routes in term of clusters from the sender to the destination. We modify a special variant of AODV routing [PBD03]. AODV itself is described in an RFC which currently is in an experimental state. Nevertheless, AODV is considered a stable and robust protocol in the ad-hoc community. [NSK03] propose an extension to make AODV multipath aware. This is the variant we use as our base for the inter-cluster routing.

When a station wants to establish a route to a destination, it sends this request to its clusterhead. The clusterhead sends out an ROUTE_REQUEST which specifies the wanted QoS parameter to all of its neighboring clusters. Every cluster checks if it has already seen this ROUTE_REQUEST, or the time-to-live (TTL) set by the source has expired. If any of these is true, the clusterhead discards the request, else, it adds its name to the path in the request. Now, the cluster checks if it would be able to handle the QoS parameters if it would be included to the route. It checks if there is enough bandwidth available, and if the delay imposed by the cluster would not outrun the requirements. If any of these is true, the request is discarded. If not, the clusterhead adds its delay to the request, so that every cluster receiving this request afterwards is able to perform the tests itself, and sends out the ROUTE_REQUEST to every neighboring cluster which is not already contained in the path.

If a cluster contains the destination, it performs the same tests as if it wanted to forward the ROUTE_REQUEST, but instead of this, it replies the ROUTE_REQUEST with a ROUTE_REPLY, containing the full path to it. Every cluster that sends a ROUTE_REPLY adds its QoS parameters to the reply, allowing the source to make a decision based on these parameters. The destination cluster answers every ROUTE_REQUEST this way, allowing the source to get every path to the destination and choose among of them which one to use. This is the main difference to AODV as explained in [PBD03], because it allows to determine more than just one path to the destination. Furthermore, it is better suited for our approach than the solution in AOMDV [MD01] which rules out many legal paths, as explained in section 3.

After receiving a ROUTE_REPLY, the source decides which path to use and enables that one by sending a ROUTE_ESTABLISH along that path. Furthermore, it chooses to use some backup routes, which are established with ROUTE_ESTABLISH along these paths, too. The ROUTE_ESTABLISH packet contains information to be able to distinguish between the main route and the backup routes.

Since more than one ROUTE_REQUEST might be pending at any time, every cluster has to remember how much bandwidth and time it has guaranteed each source. For this, the clusters not only have established resources which they use for established routes, but they also mark resources as pending. PEND_REQUEST and PEND_REPLY are the two states which are used for this. Whenever a cluster forwards a ROUTE_REQUEST, it marks the resources as PEND_REQUEST. Whenever it forwards a ROUTE_REPLY, the resource gets into the state PEND_REPLY. These two states have time-outs, ensuring that these resources are not reserved forever in case this route is not used afterwards.

Each route, regardless if it is the main route or a backup route, is monitored in order to find out if the routes break. If the main route breaks, the cluster noticing this informs every other cluster along the path. These clusters then use a detour for the information, allowing the QoS to be meet. Because of this, it is highly desirable that the chosen backup paths don't differ too much from the main route, but have many clusters in common. This allows for quick detours in case the paths are broken.

The source has to verify that the QoS parameters can be held. Because of this, it collects the information it gathers and calculates if this suffices. Furthermore, it determines the overhead of detours on the backup paths and takes these into account, too. Since the source knows if there is enough security border, and how many alternative paths are left, it calculates a *confidence value* for the route based on the number of the alternative paths and the likelihood that all paths will break at once. Thus, the application can decide if this confidence is enough for it. Furthermore, if a route breaks, the source is informed of this fact. This helps in recalculating a new confidence value. The application can be informed if the confidence value falls below an application-specific threshold. Furthermore, the source can try to find new routes once the value falls below a threshold.

Last but not least, the protocol has to cope with a changing topology. Stations move from one place to another, breaking links and creating new ones. Whenever a station moves from one cluster to another, it looses connection to his own clusterhead. If there is no other station around, the node cannot do anything and the QoS cannot be held. Anyway, if there are other stations around, the leaving node can do different things depending on what other stations it can reach: If it can reach another clusterhead, it chooses that one as new clusterhead and joins that cluster. To break ties in case that more than one clusterhead can be reached, the node monitors the network all of the time and remembers how long it can reach a clusterhead. The clusterhead which lasted the longest time will become its clusterhead, increasing the possibility that this connection will remain stable for some time.

If the station cannot reach another clusterhead, but only other stations, it forms its own cluster, that is, it becomes clusterhead itself. This is one possible case where the clusterhead becomes its own gateway.

Regardless if the leaving node joins another cluster or forms its own cluster, the new cluster tries to inform the cluster that owned the node before as fast as possible that it will be the new cluster responsible for this node. As the node will move to a neighboring cluster naturally, the old cluster can be reached very fast. Nevertheless, the time needed to inform the old cluster and to forward the information to the new one has to be taken in account when calculating the QoS parameters while the route is established.

5 Conclusion

This paper presents a new approach for routing in ad-hoc networks, allowing the utilization of QoS by establishing and exploiting fault-tolerance. The key ideas are using fault-tolerance and reducing overhead by clustering the network. Local fault-tolerance and redundancy is used to reduce restructuring the network in cases of topology changes if this can be handled in the intra-cluster routing. Again, fault-tolerance is used in the inter-cluster routing to make sure failed routes can be replaced by standby routes as soon as possible. This extra time for changing the route has been taken account for, allowing to meet the QoS even in this case. Furthermore, the source and destination are informed that the QoS parameters are likely to not hold in the future.

Literature

[BC04] Bandyopadhyav, S.; Coyle, E. J.: Minimizing communication costs in hierarchically-clustered networks of wireless sensors. Computer Networks 44(1), January 2004, pp. 1-16

[JMB01] Johnson, D. B.; Maltz, D. A.; Broch, J.: DSR: The Dynamic Source Routing Protocol for Multi-Hop Wireless Ad Hoc Networks. In: [Pe01], Chapter 5, pp. 139-172. Addison-Wesley, 2001.

[LK03] Liu, C.; Kaiser, J.: A Survey of Mobile Ad Hoc network routing Protocols. Technical Report 2003-08, Department of Computer Structures, University of Ulm, Germany, October 2003.

[MD01] Marina, M. K.; Das, S. R.: On-demand Multipath Distance Vector Routing in Ad Hoc Networks. In: Proceedings of IEEE International Conference on Network Protocols (ICNP01), 2001, pp. 14-23.

[NS03] Nett, E.; Schemmer, S.: Reliable Real-Time Communication in Cooperative Mobile Applications. IEEE Transactions on Computers 52(2):166-180. Feb. 2003.

[NSK03] Nishibayashi, Y., Sakurai, Y.; Katto, J.: Multipath Extension of AODV with Enhanced Route Establishment and Proactive Route Management. As of 2004-2-8: http://www.katto.comm.waseda.ac.jp/~katto/conference.html

[PB94] Perkins, C.E.; Bhagwat, P.: Highly Dynamic Destination-Sequenced Distance-Vector Routing (DSDV) for Mobile Computers. Computer Communications Review, pages 234-244, October 1994.

[PBD03] Perkins, C.E.; Belding-Royer, E.M.; Das, S.R.: Ad-hoc On-Demand Distance Vector (AODV) Routing. RFC 3561, July 2003.

[PC97] Park, V. D.; Corson, M. S.: A Highly Adaptive Distributed Routing Algorithm for Mobile Wireless Networks. In: Proceedings of 3rd IEEE INFOCOM '97, Kobe, Japan, April 1997, pp. 1405-1413.
[Pe01] Perkins, C. E.: Ad hoc networking. Addison-Wesley, 2001.
[Sc03] Schiller, J.: Mobile communications. 2nd edition, Pearson, 2003.
[St01] Steenstrup, M.: Cluster-Based Networks. In: [Pe01], chapter 4, pp. 75-138.