Delft University of Technology
Master’s Thesis in Computer Science

Reliable Transport for Wireless Sensor Networks

Ronald Steen

Reliable Transport for Wireless Sensor Networks

THESIS

submitted in partial fulfillment of the
requirements for the degree of

MASTER OF SCIENCE

in

COMPUTER SCIENCE

by

Ronald Steen
born in Hoorn, the Netherlands

Embedded Software Group
Department of Software Technology
Faculty of EEMCS, Delft University of Technology
Delft, the Netherlands
`www.es.ewi.tudelft.nl`

Reliable Transport for Wireless Sensor Networks

Author: Ronald Steen
Email: r.steen@student.tudelft.nl

Abstract

Wireless sensor networks (WSNs) are ad-hoc wireless networks of small form-factor embedded nodes with limited memory, processing, and energy resources. Certain applications, like security and art monitoring, require reliable data transport. Current work for WSNs only provides stochastic reliability or guaranteed reliability for bulk transfer. This thesis describes the design, implementation, and evaluation of Reliable Transport AODV (RT-AODV). RT-AODV is a protocol that provides guaranteed reliability for *individual* messages. Both routing and reliable transport are covered in this work. End-to-end positive acknowledgements with timeouts and retransmissions are used to provide guaranteed reliability. NST-AODV, an existing any-to-any routing protocol, is used to route the data and acknowledgement messages through the network. To enhance the stability and quality of NST-AODV some adjustments are done: the hardware specific Link Quality Indicator (LQI) is replaced by a statistical link estimator, and a loop detection mechanism and route quality monitoring are added. A proof of concept implementation has been done on the Embedded Software group's testbed. Experiments of larger scale and longer duration were done in TOSSIM simulations. For evaluation, Reliable Transport AODV (RT-AODV) is compared to the Collection Tree Protocol (CTP) and NST-AODV. Results show that RT-AODV performs well compared to both NST-AODV and CTP. A serious cost increase (in terms of transmissions) is involved with using end-to-end acknowledgements. The strength of RT-AODV is not pure delivery ratio. For larger networks, it performs slightly less than NST-AODV. However, RT-AODV guarantees that a message is delivered to its destination if an acknowledgement is received. The work described in this thesis shows that using end-to-end acknowledgements and any-to-any routing in wireless sensor networks is certainly viable.

Thesis Committee:

Chair:	prof. dr. K.G. Langendoen, Faculty of EEMCS, TU Delft
Supervisor:	ir. B.M. van der Doorn, SOWNet Technologies B.V.
Committee member:	dr. S.O. Dulman, Faculty of EEMCS, TU Delft
Committee member:	dr. phil. H.-G. Gross, Faculty of EEMCS, TU Delft

Preface

This thesis is the result of the work I did during the past year. I have always had an interest in programming microcontrollers and interfacing them to all sorts of things, from LEGO to fans and even deep fryers. Wireless sensor networks caught my attention when I followed a seminar by Koen Langendoen on the topic. I really liked the combination low-level hardware interfaced with radios and sensors, and hands-on programming. I got into contact with SOWNet Technologies, were I did an internship for six months. Their need for a more reliable data transport mechanism resulted in this work.

This thesis would not have succeeded without the help and interest from other people. My acknowledgements for making my internship possible –and most important, a good time– go to Bas, Michiel, and Winelis from SOWNet Technologies. Koen, my supervisor from the university, thank you for your comments and criticism, laughs, and weekly beers (or do we say Wednesday-afternoon meetings?).

This thesis marks the end of an era of my life. It took me close to a decade, and wrote part of my history. I had good fun. Thank you everyone who has been close to me during this great period. Thanks for your support, laughs, drinks, spareribs, holidays, and more. I am confident that our great times do not end here and now.

Ronald Steen

Delft, The Netherlands
6th January 2010

Contents

Preface **v**

Contents **vii**

1 Introduction **1**
1.1 Problem statement 2
1.2 Approach 3
1.3 Organisation 3

2 Background **5**
2.1 Traffic patterns 5
2.2 Routing 6
2.3 Reliable transport 7
2.4 Related work 8

3 Design **11**
3.1 Routing 11
3.2 Transport 14

4 Implementation **17**
4.1 TinyOS 17
4.2 Routing 18
4.3 Transport 23
4.4 Proof of concept 25

5 Experiments and results **31**
5.1 Simulation environment 31
5.2 Experiments 33

6 Conclusions and future work **39**

Bibliography **43**

Chapter 1

Introduction

During the last decade, the field of *Wireless Sensor Networks* (WSNs) has emerged from science fiction (Smart Dust, [17]) to a mature research topic. WSNs offer a completely new approach to many environmental monitoring and event detection applications [14, 21, 28]. This technological advance has been made possible by the miniaturisation of integrated circuits. Microprocessors, radio transceivers, and sensors, now all fit on a centimetre or even millimetre sized board.

A Wireless Sensor Network consists of small computers called nodes. Nodes are equipped with a microcontroller, a radio transceiver, sensors, and a power supply. The power supply is usually off-grid, e.g. batteries, solar power, or energy harvesting. Figure 1.1 shows an example of a typical WSN setup.

The most important characteristic of WSNs is ad-hoc network deployment; the network must organise itself without external configuration. Both the media access control (MAC) layer and the network layer (routing) suffer most from this characteristic. The MAC protocol has to decide whether a node has to sleep, send, or listen at a certain point in time. Some MAC-layer

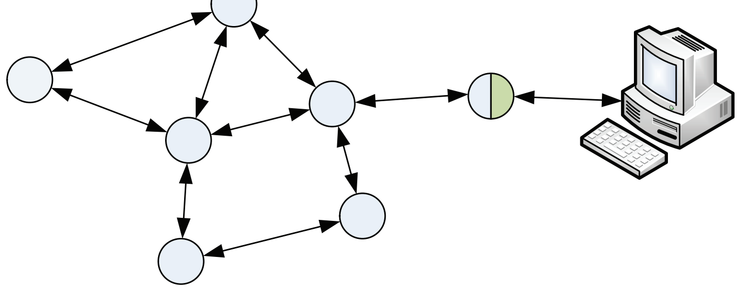


Figure 1.1: An example of a Wireless Sensor Network. The six nodes on the left are regular sensor nodes. The rightmost node is the sink, which is connected by a wired link to the outside world.

issues are node synchronisation and the hidden terminal effect. The routing protocol is responsible for forwarding messages to a next node, such that each message eventually reaches its destination. Routing protocol issues are radio-link dynamics and unknown network-topology.

In most cases, the monitoring and event detection data needs to be transported to one or more nodes called gateway or sink. These nodes are usually connected to the outside world over a long-range wired or wireless link. For the data packets to find their way through the network, some kind of routing protocol is required. In addition to best-effort data delivery, some applications require guaranteed delivery (i.e. with positive acknowledgements) for each single message. This is generally not provided by WSN protocol stacks.

1.1 Problem statement

This project started from an internship at SOWNet Technologies. First, their case is described below, followed by the problem statement it has led to.

SOWNet case description The motivation for this MSc thesis project is the need of SOWNet Technologies [1] for a *reliable data-transport mechanism* for WSNs. Their focus is the development of an art monitoring system. Guaranteed delivery of status and alarm messages to the gateway is required for such applications. Status messages are sent at a regular interval, varying from 30 seconds to 40 minutes. Alarm messages are event-triggered.

Network reconfiguration is another requirement for this application. For the reconfiguration of single nodes, it is required to be able to address individual nodes; one-to-any routing is required.

The application's network characteristics are given in Table 1.1. The application uses a two-layer network of sensing and forwarding nodes. Sensing nodes communicate over a single-hop protocol with a forwarding node. Forwarding nodes are mains powered and form a mesh network that takes care of multi-hop message transport. Sensing nodes are not an issue; this thesis only discusses forwarding nodes.

Table 1.1: Network characteristics of a typical art monitoring application of SOWNet Technologies.

topology	two layers: sensing and forwarding nodes
maximum network size	150 forwarding nodes
sensing nodes / forwarding node	20 - 100 nodes
network diameter (forwarding nodes)	20 hops
message types and interval	status (periodic), alarm (event)
maximum message delay	15 s

Problem statement Guaranteed message delivery and any-to-any routing are not supported by current transport and network protocols that are available for WSNs, see Section 2.4. Protocols that provide this functionality do exist for *Mobile Ad hoc Networks* (MANETs). These protocols, however, are not directly applicable to the problem due to limitations of WSNs. Most WSNs are constrained on available energy, radio bandwidth, memory, and computational speed.

The goal of this thesis is to develop a *reliable transport protocol for WSNs*. Previous work does not include *guaranteed delivery* or *end-to-end positive acknowledgements*. Our aim is to provide a very high delivery ratio ($> 99.99\,\%$), guaranteed delivery, any-to-one routing (data collection), and one-to-any or any-to-any routing (node reconfiguration).

1.2 Approach

This MSc thesis describes the work that has been done on the development of *Reliable Transport AODV* (RT-AODV).

First, research was performed on what current protocols and approaches are available for WSNs. None of the currently available protocols offers guaranteed delivery of individual messages or end-to-end acknowledgements. We decided to use existing work on any-to-any routing, and extend that with a reliable transport mechanism. We use a reactive any-to-any routing protocol called NST-AODV [10]. As a reliable transport mechanism we chose end-to-end acknowledgements. For each individual message, an acknowledgement is sent from the destination, back to the source. Other solutions include single-hop acknowledgements and end-to-end negative acknowledgements. However, those mechanisms do not provide the desired level of reliability. This is discussed in detail in Section 2.3.

The protocol is implemented using T-Node hardware and the TOSSIM simulator from the TinyOS distribution. The proof of concept implementation on real sensor networks hardware is used to demonstrate that the protocol can run on an actual sensor network. Then, simulation is used to evaluate RT-AODV and compare it to plain NST-AODV and the Collection Tree Protocol (CTP) [9].

1.3 Organisation

The remainder of this thesis is organised as follows. A theoretical background and related work are presented in Chapter 2. Chapter 3 discusses the functional and technical design of RT-AODV. Implementation details are given in Chapter 4. The work is verified by the experiments presented in Chapter 5. Finally, a conclusion and future work are given in Chapter 6.

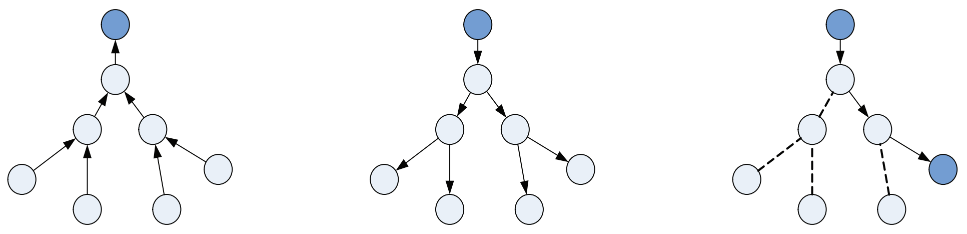
CHAPTER **2**

Background

This chapter provides a theoretical background on topics related to reliable transport in WSNs. Section 2.1 discusses the most commonly used traffic patterns in WSNs. Section 2.2 gives an overview of the different kinds of routing protocols for (wireless) ad-hoc networks in general. Section 2.3 discusses reliability mechanisms and corresponding data patterns (stream, block, single packet). Section 2.4 discusses related work on reliable transport in WSNs.

2.1 Traffic patterns

The most commonly used traffic patterns in WSNs are collection and dissemination. Collection uses an any-to-one routing scheme; all nodes forward their packets towards the same destination (i.e. the gateway). Dissemination is used to distribute a single message, injected at the gateway, over the complete network. A one-to-all routing scheme is used. Both collection and dissemination do not provide mechanisms to address individual nodes. To provide individual node reconfiguration, a one-to-any routing scheme is required. An any-to-any routing scheme would also suit the requirements. Figure 2.1 gives a graphical representation of the three traffic patterns discussed above.



(a) collection **(b)** dissemination **(c)** reconfiguration

Figure 2.1: Traffic patterns.

2.2 Routing

A routing protocol is responsible for selecting a route through a network for the messages to reach their destination. In wireless networks, and more specific, WSNs, this is hard due to frequent topology changes caused by link dynamics and failing nodes. Two important characteristics of routing protocols are link estimation and routing scheme.

Link estimation and metrics For a routing protocol to choose which route is the best one available, a quality metric is required. Hop count and expected transmissions (ETX) [4] are well known metrics. Protocols that use a metric based on hop count try to minimise the number of hops for each route. This can easily lead to the use of low-quality links, and result in unreliable routes.

The goal of ETX-based protocols is to minimise the number of expected transmissions for each route. Each node maintains link-quality statistics for a small set of neighbours about the incoming and outgoing link quality. The incoming link quality can be determined directly based on the ratio of received an missed beacon messages; those messages are transmitted at an exponentially increasing interval. Incoming link qualities have to be exchanged with neighbouring nodes to determine their outgoing link qualities. The outgoing link quality from node A to node B is equal to the incoming link quality from node B to node A. The incoming and outgoing link qualities together form the bidirectional link quality. It is important to use bidirectional links. If a unidirectional link is used, link layer acknowledgements are not received by the sender; this can cause the sender to repeatedly retransmit a message without receiving an acknowledgement.

Routing schemes Routing protocols can be classified in three schemes: proactive, reactive, and hybrid.

Proactive routing schemes maintain routing information for the complete network. Each node has a routing table with an entry for each other node. An entry consists of a destination, the next hop to reach that destination, and route information. Routing information is updated periodically, or triggered by network topology changes. Some examples are the *Destination-Sequenced Distance-Vector* (DSDV) routing protocol [24], and the *Optimized Link State Routing* protocol (OLSR) [3]. DSDV uses hop-count as a routing metric. For each destination, a sequence-number determines the freshness of a route. OLSR determines for each link whether it is bidirectional or not. Only bidirectional links are used for routing. OLSR uses an optimised flooding mechanism to distribute routing information over the network.

Reactive, or on-demand routing schemes maintain routing information for active routes only. If a route to the corresponding destination is not available at the current node, a *route discovery process* is started to retrieve the required information. Two kinds of reactive routing are available: source routing and hop-by-hop routing. In protocols using source routing, each packet carries its complete route to the destination. Intermediate nodes forward the packet over this path. An example is *Dynamic Source Routing* [16]. Hop-by-hop routing protocols only include the packet's destination. Each node forwards the packet over the route that is present in its routing table. An example of a reactive hop-by-hop routing protocol is *Ad hoc On-demand Distance-Vector* routing (AODV) [23]. NST-AODV [10] is an implementation of AODV developed for the sensor networks platform TinyOS [20]. Implementation details of, and adaptations to NST-AODV are discussed in detail in Chapter 4.

A routing scheme that uses both reactive and proactive mechanisms to maintain routing information is called a hybrid routing protocol. Most hybrid protocols use some form of zone routing. A node maintains routing information in a proactive fashion within its zone. Information about routes to nodes outside its zone are maintained in a reactive fashion. The *Zone Routing Protocol* (ZRP)[11] is an example of a hybrid routing protocol.

2.3 Reliable transport

A transport protocol provides message delivery on top of a routing protocol. Possible features are message ordering, reliable message delivery, and congestion control. The focus of this thesis is on reliable message delivery; therefore this section only discusses reliable transport.

Reliable transport protocols can provide two types of reliability: stochastic- and guaranteed reliability. Stochastic reliability promises to deliver a message with a certain probability. To provide stochastic reliability, or best-effort delivery, most protocols do a limited number of single-hop retransmits, or send the message over multiple routes. Stochastic reliability is too weak to fulfil the requirements for the art monitoring application given in Chapter 1.

Guaranteed reliability uses an acknowledgement mechanism to let the source node know whether a packet has been received at the destination, or is missing. To provide guaranteed reliability, the source node of a message requires feedback from the destination. A distinction is made between messages that are part of a block or stream, and messages that are not.

For messages that are part of a block or a stream, it is possible to detect the absence of a message at the destination, and it suffices to send negative acknowledgements for missing messages. For a message block, the block size is known by the destination. For a message stream, a later message will be received, which indicates that previous messages are missing.

On the other hand, for messages that are not part of a block or a stream, the destination has no knowledge about a missing packet that has not been received. In this case, positive acknowledgements and a timer at the source are required for each message to achieve guaranteed reliability.

For both positive and negative acknowledgements, the routing layer must provide means to send acknowledgements from the destination back to the source.

2.4 Related work

For typical WSN applications, data collection is the most important traffic pattern. An environment is continuously monitored for events, or periodic measurements are done. This data has to be reported to the base station. In most cases, receiving the majority of the data is most important. Gaps that are caused by failed messages can either be left open, interpolated, or a retransmit can be requested. For these cases stochastic reliability or guaranteed bulk-transport are sufficient.

Most WSNs are battery powered, and therefore energy constrained. A trade-off in energy efficiency, reliability, throughput, routing scheme, network lifetime, and message delivery delay is required. For the above-mentioned applications, energy efficiency can be traded for a reliability scheme that has relatively low overhead, like guaranteed bulk-transport or stochastic reliability. A simple routing scheme that only allows to forward messages to a base station is sufficient in most cases. Each application has its own throughput and delivery delay requirements. The limited hardware platform, applications, and requirements have led to the development of several routing and transport protocols. The protocols and protocol classes that are most relevant to our work are summarised below.

The Collection Tree Protocol The Collection Tree Protocol (CTP) [9] is the default collection protocol of the TinyOS distribution. A network that uses CTP contains one or more base stations, or root nodes. Root nodes advertise their presence to neighbouring nodes. Non-root nodes advertise their distance to a root in terms of the expected number of transmissions (ETX). This way, each node maintains a route to the closest root node. Loop detection and topology repair are well handled. The results presented in [9] show delivery ratio's of over 90 %. Our experiments, as discussed in Chapter 5, show much lower delivery ratio's for network sizes of over 20 nodes.

Reliable bulk-transfer protocols Some examples of protocols that provide reliable bulk-transfer are PSFQ [27], FLUSH [18], and RCRT [22]. These protocols determine if all messages are received at the destination. When a message is missing, a negative acknowledgement is sent to the source, and the message will be retransmitted.

TinyHop TinyHop [25] is a reactive routing protocol that provides end-to-end acknowledgements for individual messages. The metric that is used to determine which route to use is based on time. Each node only forwards the first route discovery message for each individual route request. The route of the message that arrives first at the destination is used. This method can easily lead to the use of unstable routes, or routes with a low delivery ratio. Experiments show low delivery ratios, especially for larger networks (128 nodes, 20 hop diameter, delivery ratio $< 20\,\%$).

Chapter 3

Design

The hardware platform for which Reliable Transport AODV (RT-AODV) is developed has a grid-powered forwarding network; it is not energy constrained. With respect to current work, this allows us to provide a higher level of reliability, and a more flexible routing scheme. The applications that RT-AODV is designed for require guaranteed delivery of individual messages. The only way to provide this level of reliability is by end-to-end positive acknowledgements.

We propose to use an any-to-any routing protocol; the rationale of this solution is twofold: it allows to route acknowledgement messages back to the source, as well as sending reconfiguration messages to arbitrary nodes. We have built our solution on top of NST-AODV [10], a reactive, any-to-any, hop-by-hop routing protocol. We have improved the original implementation of NST-AODV by making the route discovery mechanism more robust, adding loop detection and loop handling mechanisms, and adding a statistical bidirectional link-estimator. This work brought NST-OADV to a level of stability such that it is usable for commercial applications. On top of the adapted routing protocol we have implemented an end-to-end positive-acknowledgement mechanism with timeouts and retransmissions. Figure 3.1 gives an overview of the proposed protocol stack, an overview of the subcomponents of NST-AODV, and the internal information flow of NST-AODV.

This chapter describes the design of RT-AODV. Section 3.1 discusses the used routing protocol (NST-AODV), and the working of its components. Section 3.2 covers the design of the transport mechanism.

3.1 Routing

Routing is handled by NST-AODV, which is a reactive, hop-by-hop, any-to-any routing protocol, specifically designed for WSNs. It provides the necessary mechanisms for route discovery, route maintenance, message forwarding, and link estimation. Figure 3.1 gives an overview of the components of NST-AODV, and the information flow between the components.

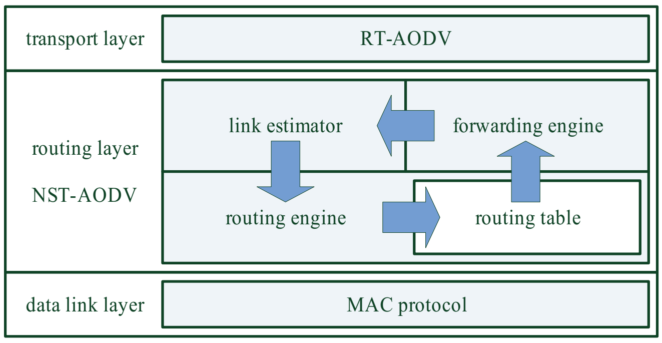


Figure 3.1: Protocol-stack overview of RT-AODV with subcomponents of NST-AODV. The arrows indicate the internal information flow of NST-AODV.

The routing engine maintains the routing table. There is one entry for each destination. An entry contains the following information: destination address, route sequence number, next-hop address, route length, and route quality. If no route is available for a required destination, the routing engine starts a route-discovery process to find one.

The link estimator provides statistical information about the quality of individual links. The quality of a route is determined by the product of the quality of each link on that route, see Equation 3.1.

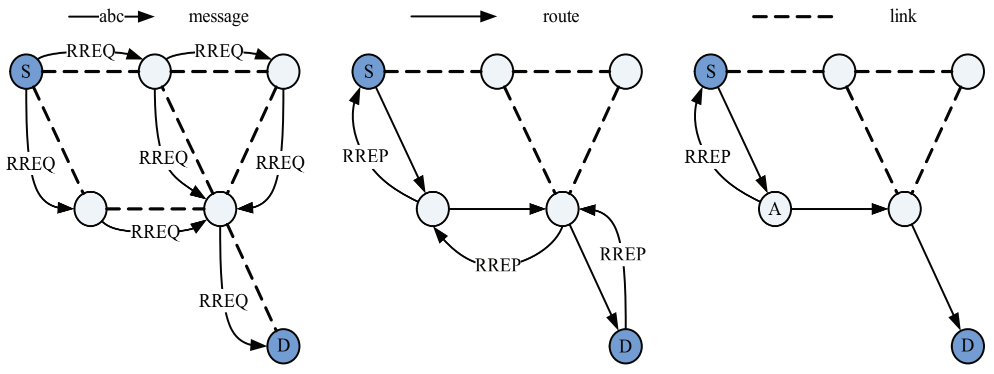
Messages are forwarded by the forwarding engine. For each message, the next hop to reach the destination is taken from the routing table. One of the major concerns for routing protocols are routing loops. Due to the decentralised approach of finding and maintaining routes, it is difficult to prevent loops. The forwarding engine is responsible for detecting routing loops.

$$quality\,(reception\,ratio)_{route} = quality_{link\,1} \times \ldots \times quality_{link\,n} \tag{3.1}$$

3.1.1 Route discovery

When a node wants to send a message to a destination, the routing table is searched for a corresponding route. If a route to the destination is unavailable, a route-discovery process is started. A route-request message is flooded through the network until it reaches the destination. Each node records the current information contained in the route-request message. A route-request message contains the following information: source address, destination address, source sequence number, last-known sequence number of the destination, current route length, and current route quality.

If a route-request message reaches its destination, a route reply message is sent back to the source over the route that the route request had travelled to reach the destination. Upon reception of a route reply, a node checks its



(a) The route request is flooded through the complete network.

(b) A direct route reply: the destination (node D) sends a route reply.

(c) An indirect route reply: node A already has a route to the destination (node D) and sends a route reply immediately.

Figure 3.2: Direct and indirect route discovery processes.

previously recorded information, which includes the next hop to the source, and adds the new route to its routing table.

For a faster response to a route request, and less overhead, indirect route discovery is allowed; nodes that have a valid route to the destination can also send a route reply. Figure 3.2 explains the difference between a direct and an indirect route discovery process.

Each route discovery process is uniquely identified by its source address, source sequence number, and destination address. If a route request is received, processing and forwarding is only carried out for newer or better routes. A newer route request is recognised by a higher sequence number. If the sequence numbers of the last and current route request are equal, the last route request is only accepted if its route quality is higher. Route replies are selected analog to route requests. As a result, the route with the best quality is recorded in the routing table.

3.1.2 Link quality estimation

The link-quality estimator that we included in RT-AODV is the implementation that is also used for CTP [9]. It uses information from periodic beacons, as well as data messages. Beacons are broadcasted by each node. Each beacon includes a sequence number and data entries of the *Link Estimation Exchange Protocol* (LEEP) [8]. After a fixed number of beacons has been received from one node, the sequence numbers are used to determine the number of missing beacons. The forwarding engine reports for each unicast message whether or not an acknowledgement has been received.

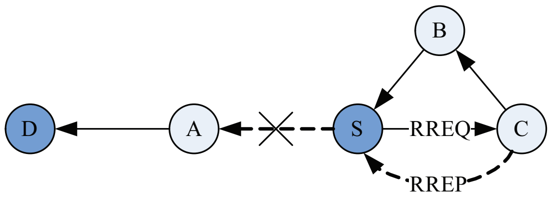


Figure 3.3: An example scenario that results in a loop. The link from node S to node A is part of the route from node S to node D. This link fails, and node S initiates a route discovery. A route request (RREQ) travels from node S to node C, which node C replies to (RREP) with its available route via node B to node D. The route from node C to node D includes node S, and will form a loop. Based on their local information, both node S and node C cannot determine that a loop is being formed.

The estimates based on beacons are usually very rough. However, this estimate is only used as an initial value, and will become more precise when the link is actually used to forward data messages [9].

3.1.3 Route quality estimation and evaluation

After a route has been determined, the actual quality of the route may vary over time. This variation can be caused by environmental changes and network utilisation. In order to deliver all messages successfully, we try to notice the symptoms before failures happen.

If the delivery ratio of a route drops below a threshold, a route discovery process is started to try to find a better route.

3.1.4 Routing loops

Due to the decentralised routing approach generally used in WSNs, loops will likely form at some point in time. An example of how indirect route-discovery can lead to a routing loop is given in Figure 3.3.

A loop is identified if a message passes the same node twice, with a time-to-live (TTL) that differs at least two hops from the previous reception. If a loop has been detected, a process is started to remove the conflicting entries from the routing table.

3.2 Transport

A given fact about wireless data transmission is that intermittent failures occur [26]; a mechanism that handles those failures is required. We have designed and implemented an end-to-end positive acknowledgement mechanism; this is the only type of mechanism that can determine that an individual message has been successfully received at the destination. If an acknowledgement is not received within the timeout period, two things might have happened: the data message did not reach the destination or the acknowledgement message did not reach the source. In both cases, the only way to request a new acknowledgement is to retransmit the data message from the source.

If there is no maximum number of retransmissions, the transport protocol could try to deliver a message that continuously fails. To prevent this, a limited number of retransmissions is allowed. If this limit is reached, the message is flooded through the complete network as a final attempt. If the message is successfully received at the destination, an acknowledgement is sent back to the source in the usual fashion.

Flooded messages contain the same route discovery information as route-request messages do. Each node that receives a flooded message will record the route information, as it would do with a regular route-request message. If the message reaches its destination, and the route satisfies the minimum quality, a route reply is sent to the source.

Chapter 4

Implementation

For the implementation of our reliable transport protocol we use NST-AODV as the underlying routing protocol. The original implementation of NST-AODV is not very robust and lacks important features like a statistical link-estimator, loop detection, and route monitoring. We have made improvements to enhance robustness, and made additions to include the required features. All improvements and additions are discussed in the remainder of this chapter. The implementation is done in TinyOS, a commonly used operating system for WSNs. We speak of NST-AODV as the routing protocol, and of RT-AODV as the transport protocol on top of it.

This chapter provides an in-depth description of the implementation details of RT-AODV. Section 4.1 gives a short introduction to TinyOS 2.0. Section 4.2 provides details about the routing protocol like the route discovery mechanism, message-header structure, and the link estimator. It covers work that is part of the original implementation; this is useful for a good understanding of our own work. Section 4.3 discusses the implementation of the end-to-end acknowledgement and retransmission mechanism. As a proof of concept, we have implemented RT-AODV on a sensor network hardware platform. This is discussed in Section 4.4.

4.1 TinyOS

TinyOS [20] is an event-driven operating system for WSNs. It takes care of low-level hardware interfacing and scheduling. The programming language for TinyOS is nesC [7], which is a derivative of C. Programs in nesC are defined as *components* and *implementations* of those components. Components are *wired* together by *int*erfaces; an interface defines *commands* and *events*. The user of an interface can call commands, and is notified of each firing event. TinyOS provides hardware support for a wide range of platforms and components. It also provides platform independent components and commonly used software components, such as timers, data structures, radio stacks, etcetera.

TinyOS also includes TOSSIM [19], which is a WSN simulator. The simulator can run the exact same code as the corresponding hardware platform does. For our simulation implementation we have modeled the lnode platform [1]. It simulates an MSP430 series microcontroller and a CC1101 packet radio.

4.2 Routing

This section describes the components of NST-AODV. First, the message structure is discussed. Then, each component is discussed, and the fields that are used and modified by it.

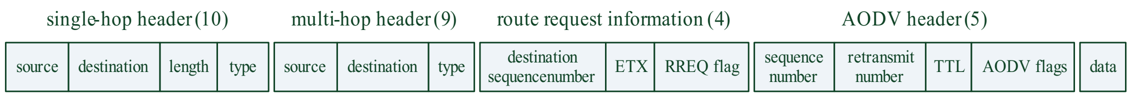
4.2.1 Message structure

Figure 4.1 gives a graphical representation of the NST-AODV control and data messages. The single-hop header fields are used for single-hop transmission by the radio. It is provided by the *Active Message* abstraction of TinyOS [2]. NST-AODV uses four message types: route request, route reply, route error, and data. The message type is indicated in the type field of the single-hop header.

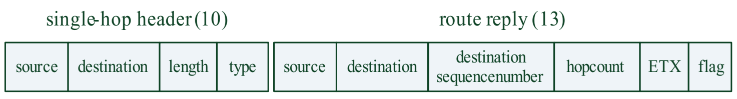
A data message contains a single-hop header, a multi-hop header, and an AODV header. The multi-hop header contains the multi-hop source and destination, and the type of the data message. These are used by the forwarding engine.

The AODV header contains route-request information, sequence number, retransmit number, time-to-live (TTL), and AODV-specific flags. The sequence number, retransmit number, and the multi-hop source address denote a unique message. Those values are used for duplicate filtering, loop detection, and to identify corresponding acknowledgements and messages. The route-request information part carries the sequence number of the last known route to the destination, the current ETX value, and route-request specific flags. If a message is a route request or a flooded message that contains route-request information, the number of hops equals $MAX\ TTL - TTL$. The route-request flags indicate whether or not an indirect route reply is allowed. A route-request message has the same structure as a data message, but does not contain any data. NST-AODV uses 16 or 32-bit addresses, depending on the platform. For the lnode platform, which uses 32-bit addresses, NST-AODV introduces 19 bytes overhead per message. This excludes the 10 byte single-hop header that is required by any routing protocol. Given a maximum packet size of 64 bytes, there are 35 bytes payload left.

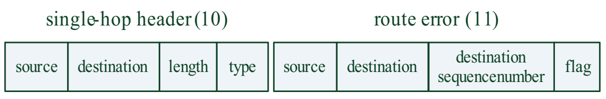
A route-reply message contains the usual single-hop header, the source address and destination address of the route, the sequence number of the route, route length, route quality in ETX, and route reply flags. The flags indicate that the reply is direct (i.e. from the destination) or indirect (i.e. from an intermediate node).



(a) route request and data message



(b) route reply message



(c) route error message

Figure 4.1: Structure of NST-AODV message types. Size in bytes is given between parentheses. 32-bit addresses are used.

The route-error specific fields are source, destination, sequence number, and flags. The destination and sequence number identify the route that has to be removed from the routing table. The source address is included to stop forwarding the route error message when it reaches the source of the route error.

4.2.2 Routing engine

The routing engine covers the mechanisms for route discovery, sending and processing of route error messages, and routing table maintenance.

Route discovery The route-discovery algorithm consists of two stages: flooding a route request through the network, and sending a route reply from the destination back to the source.

Each time a node forwards a route request, the hop count and route quality contained in the route request are updated. The number of hops is increased and the ETX value of the last hop is added to the route ETX. For link ETX, route ETX and route length, maximum values are determined. Routes for which those maximums are exceeded, will very likely be unreliable, or needlessly inefficient; those route-request messages will not be forwarded.

All nodes maintain a cache of recently-received route discovery information. If a route request has been received, a look-up for the source address and sequence number is done. If an entry exists, the route request was not the first one received. If two route requests originate from the same route discovery, they are compared by their ETX value. Only route-request messages that correspond to a new route discovery process, or to routes that are of better quality than the current best route, are processed and forwarded. A route-request message carries a flag that indicates that direct route discovery is compulsory or not.

During a route discovery for a certain destination, messages that are addressed to that node are put on hold. However, the node can still continue forwarding messages to other destinations.

Sliding window mechanism The original implementation of the routing protocol is not resilient to sequence-number overflows and rebooting nodes. In both cases, control messages are recognised as out of order, and are discarded.

This is easily solved by a sliding-window mechanism. We define the window of valid sequence numbers as all sequence numbers that are newer or at least the window size older than the last sequence number. All sequence numbers that are at most the window size smaller are in the window of invalid sequence numbers. Only messages that have a valid sequence number are accepted, others are discarded.

After a node has booted, its sequence number is initialised to 0. If its route requests are not replied to, its sequence number is possibly in the window of invalid sequence numbers. To overcome this, a node increases its sequence number by the window size. This is repeated until a route reply is successfully received.

Routing table In WSNs, most traffic is directed to and from a single node: the base station. As the network fans out quickly, nodes that are further away from the base station forward traffic for fewer nodes, and also need to maintain fewer routes.

To minimise control overhead, the routing table needs to have the size of the number of nodes in the network. This can become a problem for large networks. However, increasingly more memory is available within the same power envelope. A possible solution to minimise routing table size for large networks will be discussed in Chapter 6.

Route-quality monitoring The original routing protocol does not monitor routes that are currently in use; actions are taken only if a route is no longer available. To achieve a delivery ratio that is as high as possible, we do not want a route to suddenly stop working. Therefore, we have implemented a route-quality monitoring system.

We distinguish three situations: a failing forward route, a failing reverse route, and a node that has difficulties in finding a route. Detection of the symptoms of each situation takes place at the destination. To filter high frequency, or short term changes, an exponentially weighted moving average is used.

Problems on the forward route lead to a high number of transmissions that is needed before a message is successfully delivered to the destination. This is indicated by a high average retransmission number of the message that is received first. The destination informs the source by setting a flag in the AODV-flags section of the acknowledgement message. The source responds by starting a direct route discovery to find a new route.

In case of a failing reverse route, messages on the forward route are received successfully, but acknowledgements are not received by the source. This is indicated by a high number of received retransmits per single message. In this case the destination node starts a direct route discovery to the source.

In the first two situations, the alternative route that results from the route discovery only replaces the current route if it is of good quality. The quality of the current route is very likely not representative for the current situation; we cannot use it to compare the current and the alternative route. Therefore, it is not required that the new route is of a better quality than the current route.

The third situation covers nodes that have difficulties in finding a route to the destination. In this case a node floods its messages to the destination. If a high ratio of messages from a node is flooded, this indicates that the node can probably not find a valid route and has connectivity problems. This can be indicated to the application; an engineer can then respond by adding additional nodes or move the node to have better connectivity.

4.2.3 Forwarding engine

The forwarding engine transmits messages to the next hop and detects if a problem occurs. Problems like routing loops and link failures are signalled to the routing engine.

Loop detection and handling Loop detection takes place at two locations: at forwarding nodes and at source nodes. Forwarding nodes use a recently received message cache to detect loops. If the cache contains an entry that corresponds to the source address and sequence number of the received message, and the TTL differs by two or more hops, the message has likely travelled over a loop. Source nodes detect loops by checking a message that is received to be forwarded. If the source address equals the local address, this indicates a loop.

If a loop is detected, a route-error message is sent over that route. The message is forwarded at each node until it reaches the original node that detected the loop. In this case, the route error traversed the complete loop. A loop can also be temporary; it can disappear before the route error has traversed the complete loop. In this case, the route error will reach the destination of the route, and will not be forwarded as well. If a route error is received by a node, the node removes all routesthat travel over the loop from its routing table.

To decrease the probability that the next route discovery process results in a loop, the initiated route-discovery process does only allow direct route discovery.

Link failures If for a single message all single-hop retransmissions fail, i.e., no link-layer acknowledgement is received, the current link-quality is evaluated. If the quality drops below the threshold, all routes from that node with the next hop that is equal to the destination of the failing link, are removed. A route discovery process will be started when one of the removed routes is required at a later time. No further action is taken at the forwarding node; end-to-end retransmissions will resolve single-hop failures.

4.2.4 Link-quality estimator

The original implementation of NST-AODV does not include a statistical bidirectional link-quality estimator; the *Link Quality Indicator* (LQI) value of the CC2420 radio [15] is used to determine route quality. We have replaced the LQI value by the link estimator that has been developed as part of CTP for two reasons: LQI does not provide a very good estimation of the actual link quality [6], and it is only available on specific radio hardware.

The metric that is used to evaluate route quality is the expected number of transmissions (ETX). Internally, the link estimator uses both ETX and reception ratio to represent link quality. ETX is expressed in terms of reception ratio in Equation 4.1. To determine the quality of a route, Equation 4.2 is used.

$$ETX = \frac{1}{reception\,ratio} \tag{4.1}$$

$$ETX_{route} = ETX_{link\,1} + ETX_{link\,2} + \ldots + ETX_{link\,n} \tag{4.2}$$

The link estimator determines link quality based on the reception ratio of beaconing messages and data messages. The beaconing messages are used to determine an initial estimate. The estimate is made more precise when the link is in use to transmit data messages.

Beacon messages are sent periodically. The interval increases exponentially from 2 seconds to 5 minutes. Nodes that have recently joined the network or have no valid route available can set the *pull bit* in their beacons. If a neighbouring node receives a beacon with a pull bit, it resets its interval to 2 seconds. This allows for faster bidirectional link estimation at node start-up.

In addition to the beacon-based estimate, the reception ratio of data messages is used to obtain a more precise estimate of the link. The link layer reports to the link estimator whether or not a link-layer acknowledgement was received for each message.

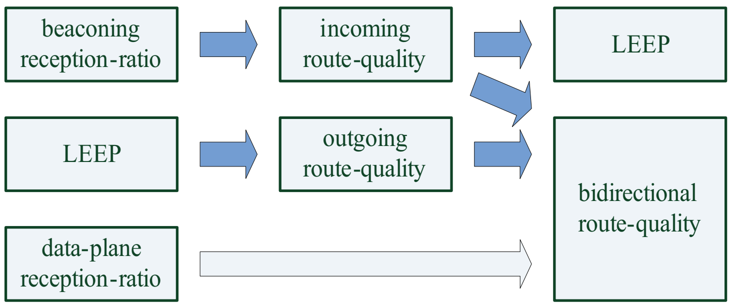

beaconing reception-ratio
incoming route-quality
LEEP
LEEP
outgoing route-quality
data-plane reception-ratio
bidirectional route-quality

Figure 4.2: link-quality estimation “data-flow”

Beaconing only allows a node to determine the incoming quality of a link. To calculate bidirectional link-quality, nodes exchange incoming link-quality information using LEEP [8]. The LEEP protocol piggybacks link quality information on the beacon messages. Equation 4.3 is used to calculate the bidirectional link-quality from the incoming and outgoing link-quality. The ratio of the number of acknowledged messages to the total number of messages directly represents bidirectional link-quality. Each time the bidirectional link-quality is calculated, the outcome is fed through an exponentially weighted moving average. Figure 4.2 gives an overview of how link-quality information flows through the link estimator.

$$quality_{bidirectional} = \frac{1}{quality_{incoming} \times quality_{outgoing}} \tag{4.3}$$

4.3 Transport

This section describes the implementation details of the transport layer. The transport layer is implemented on top of the routing protocol that has been discussed in the previous section. The transport layer implements an acknowledgement and retransmission mechanism to provide guaranteed message delivery.

Each data message carries a sequence number and a retransmission number in the AODV header (Figure 4.1a). An acknowledgement message only contains the sequence number of the corresponding message; a delayed acknowledgement message corresponding to the first transmission will also acknowledge a later retransmission.

For a message to be successfully delivered, at least the following three events have to take place: the message needs to be enqueued at the source, the source needs to successfully send the message to the next hop of a valid route, and it needs to be acknowledged. Those events can fail and happen out of order. The state machine given in Figure 4.3 handles the state transitions of end-to-end messages.

Acknowledgements and forwarded messages that originate from another source are not kept in the queue to wait for an acknowledgement. They are removed as soon as a *sendDone* event has occurred. In TinyOS, a sendDone event is fired when the radio is done transmitting a message and is ready to accept the next message.

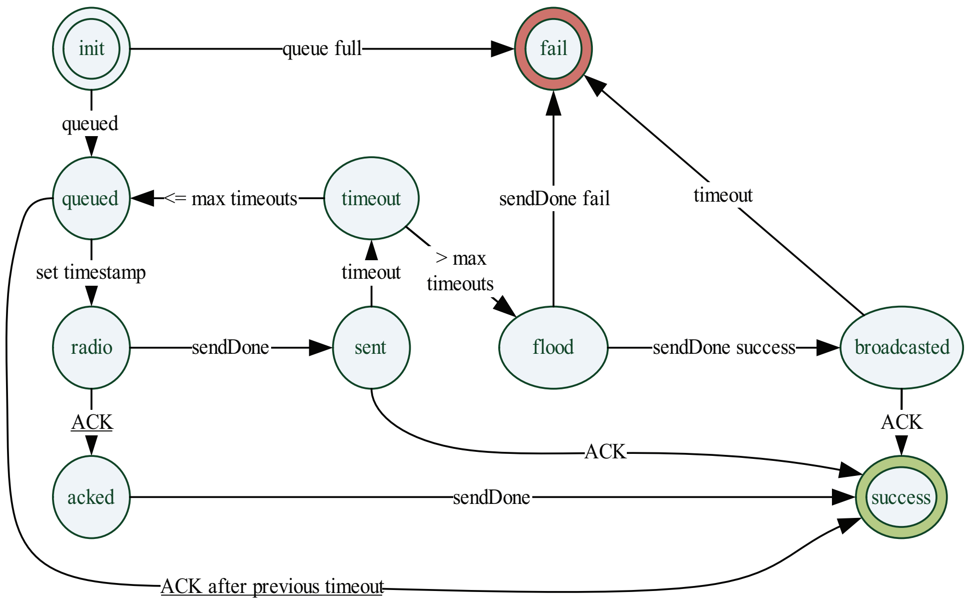


Figure 4.3: State machine. The underlined acknowledgement events happen out of order.

Timeouts The current implementation uses a predefined static timeout period. It is not dependent on the length of a route, or the average round trip time of the route. It is required to determine a suitable value for each network, such that the most remote nodes are able to deliver their messages within the timeout period. In the future, the timeout period can be determined as a factor of the moving average of the route trip time. In case of a network-wide emergency flood, the timeout period is doubled to increase the probability that the acknowledgement is received on time. A typical timeout period is in the order of seconds. We poll for timeouts twice every second. A timeout occurs if an end-to-end acknowledgement is not received on time.

Retransmissions If the maximum number of retransmissions has been reached, or a timeout occurred after a network wide flood, the transport mechanism has failed to deliver the message. How this situation is handled is up to the application.

Out-of-order events Two situations occur in which an acknowledgement is received out of order, i.e. not right after a successful *sendDone* event; see Figure 4.3. A single-hop message can be successfully transmitted to the next hop while the link-layer acknowledgement has failed. The source node is still trying to deliver the message to the next hop while the message has already reached its end-to-end destination An acknowledgement is sent back and is received at the source before the node knows that is has successfully transmitted the message to the next hop. After the acknowledgement has been received we have to wait for the single-hop transmission to finish, and the radio to reach its initial state. Then we can send the next message.

The second situation involves the reception of an acknowledgement that corresponds to a previous end-to-end transmission that has already timed out. This acknowledgement has a different retransmit number than the last retransmit number, but the corresponding sequence number will acknowledge the message.

4.4 Proof of concept

This section discusses the implementation of RT-AODV on sensor network hardware. The prototype implementation has been done on the testbed of the Embedded Software group of Delft University of Technology. The testbed consists of 20 T-Nodes that are placed in four rooms [13]. A T-Node consists of an ATMega128L, 8-bit, 8 MHz microcontroller with 128 kB of programmable flash and 4 kB of RAM, and a CC1000, 868 MHz, 76.8 kBaud radio. Figure 4.4 shows the network topology of the testbed.

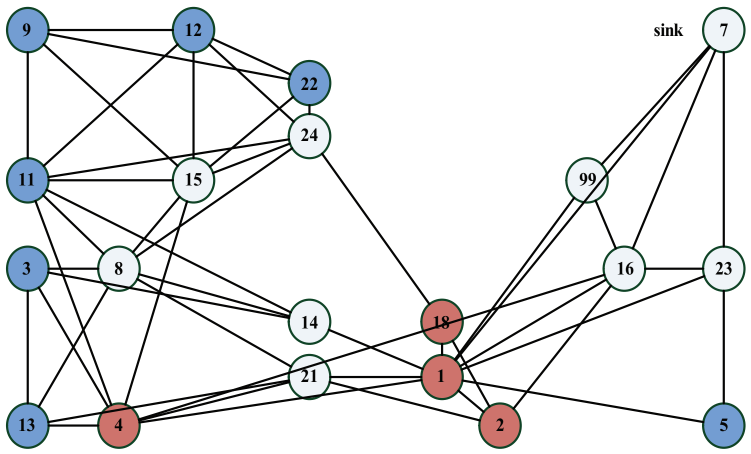


Figure 4.4: Testbed topology. Each line represents a link with at least 90 % bi-directional delivery ratio. Dark blue nodes periodically send a message to the sink. Red nodes are shut down one by one each hour in experiment 2.

Table 4.1: The memory footprint of RT-AODV components, based on 16-bit addresses.

component	number of entries	entry size (B)	RAM (B)
routing table	12	20	240
data message queue	8	65	520
control message queue	8	52	416
flood cache	12	7	84
route-request cache	12	13	156
total			1416

Table 4.1 shows the memory footprint of RT-AODV per component. RT-AODV requires more memory than typical WSN collection protocols, e.g. CTP maintains an 80 byte routing table of 10 entries of 8 bytes. This is due to the fact that we have to maintain a much more complicated routing table than tree-based protocols, which only transmit messages in the direction of the sink, and not back to the original source; this requires only a single route.

4.4.1 Experiments

The experiments evaluate RT-AODV under three different circumstances: a stable network, a network with failing and later restarting nodes, and a network with radio jamming. We have implemented a simple TinyOS program that reports the node's supply voltage every 5 minutes for the six blue nodes in Figure 4.4. All nodes forward messages to the sink, node 7.

Experiment 1: normal conditions

This experiment has been carried out under normal conditions. The results given in Figure 4.5 show that the network is very stable, and few end-to-end retransmissions are required. All messages have been delivered successfully.

Not all messages were acknowledged after the first transmission. If we count those messages as failed messages, the delivery ratio is 99.83 %. In this experiment, RT-AODV gains 0.17 % successful delivery by including end-to-end acknowledgements.

Experiment 2: joining and leaving nodes

This experiment is done with a network that contains joining and leaving nodes. Some of the sending nodes (blue) require one or more of the failing nodes (red) to deliver messages to the sink; see Figure 4.4. When one of these nodes is shut down, routes that currently flow through it will fail. This triggers the route discovery process to reconnect the network. The results

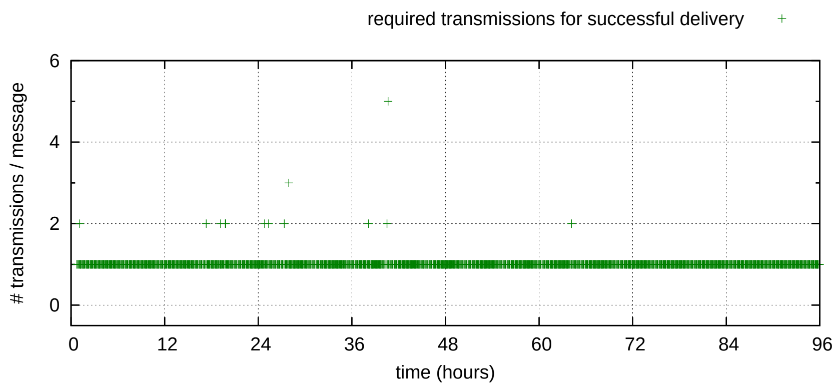
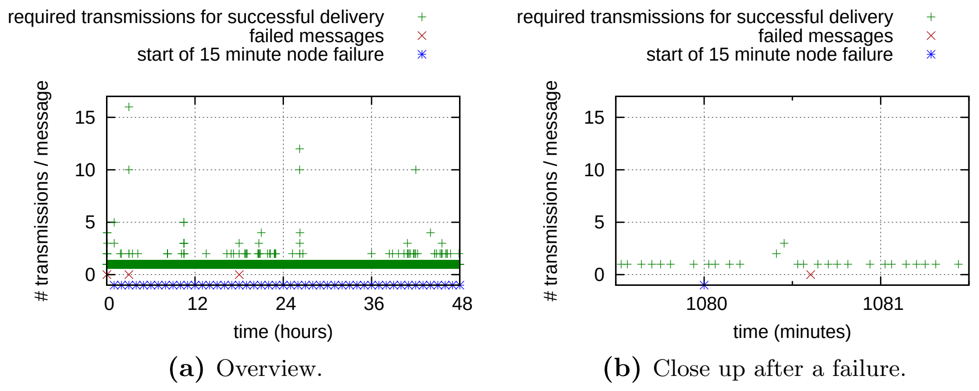


Figure 4.5: Experiment 1: normal conditions

given in Figure 4.6 show very few end-to-end retransmissions. With respect to *experiment 1*, we observe a small increase in the number of required end-to-end transmissions. Most of the time retransmissions are bursty, but there is no strict grouping right after a node failure.

Three messages, out of 52 230, have failed during the two day experiment. The cause of the failures is not very clear. An explanation that relates to the joining and leaving nodes is that a leaving node broke a route that could not be repaired in time. In the testbed network, a joining node can trigger almost the complete network to reset their beaconing period. The large number of beacons can interfere with the propagation of route discovery messages. This can result in bad link quality estimates and failing route requests.

This experiment shows that RT-AODV is resilient to node failures and topology changes. If changes in the network occur, RT-AODV stabilises quickly on a new route.



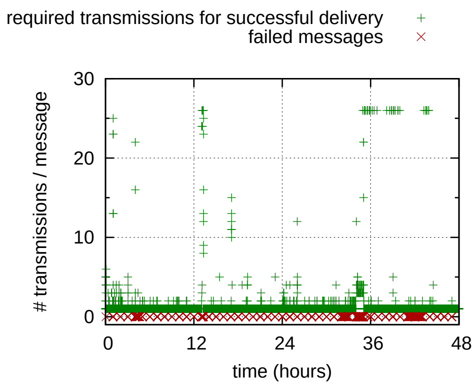
(a) Overview. **(b)** Close up after a failure.

Figure 4.6: Experiment 2: joining and leaving nodes. Each 60 minutes a node is shut down for 15 minutes.

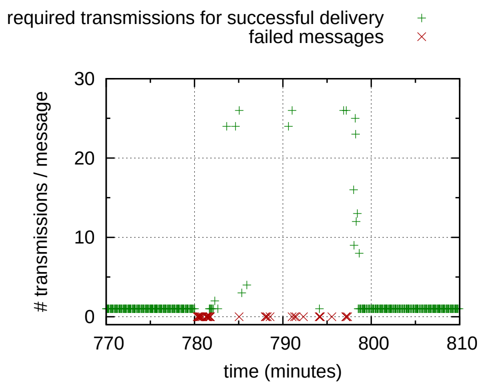
Experiment 3: network jamming

This experiment covers network jamming. One node of the testbed is programmed to transmit a large number of messages at a much higher power level than the regular nodes. This makes that the other nodes cannot communicate over their radios during that period. Similarly to experiment 2, this experiment also involves triggering of route discovery and reconnecting the network. The difference with experiment 2 is that nodes are not rebooted, and thus do not return to their initial state after the communications failure is over.

During the 15 minute jamming period, a lot of messages fail to reach the sink. The nodes only try to forward a message for a short period of time ($\#retransmits \cdot timeout\ period = 25 \cdot 6 = 150\,\mathrm{s}$). As we can see in Figure 4.7b, the network returns to a stable state within 4 minutes after the jamming period is over, at 795 minutes.



(a) Overview.



(b) 40 minute closeup of jamming at 13 hours.

Figure 4.7: Experiment 3: one jamming node that is turned on every hour for 15 minutes.

Overall experiment results

The overall results of the three experiments are given in Table 4.2. Overhead is calculated as the ratio of the number control or beaconing messages to the number of data messages. The results show that it is feasible to implement and deploy RT-AODV on a sensor networks hardware platform. It can deliver messages with a high probability, very close to 100 %. Failing and joining nodes and network jamming both cause some failed messages; however, the protocol still performs good.

The overall results show an increase in control overhead for the node failures experiment; compared to the normal experiment the control overhead quadruples. Failing and joining nodes cause routes to fail and new routes to form.

Table 4.2: Experiment results of the proof of concept implementation.

	normal	**node failures**	**jamming**
delivery ratio (%)	100.000	99.998	99.851
control overhead (%)	0.14	0.59	7.53
beaconing overhead (%)	5.77	14.79	4.76
average hops	2.85	3.06	3.39

This results in a fair increase of control overhead. Network jamming causes a much bigger increase in control overhead. When the complete network is being jammed, all links fail, and routing table entries are marked invalid. For each route, a new route discovery has to be started each hour. In normal situations, a route requires very little maintenance.

As we can also see in the overall results, joining nodes introduce a lot of extra beaconing overhead; beaconing overhead almost triples compared to the normal experiment. This is caused by the pull bit that is set in the first couple of beacons that a node broadcasts after starting the routing protocol. Due to very high connectivity in the testbed, messages will be received by a large part of the network. It causes a node that receives such a beacon to reset its beaconing interval to two seconds. The jamming experiment has less beaconing overhead compared to the normal experiment. However, this difference of almost 1 % is much smaller than the difference between the normal experiment and the node failure experiment.

The average number of hops that is required to reach the destination can be used as a measure for route quality. A shorter route, that is still reliable and stable, is better than a longer one. The differences between the three experiments are not very large. We can conclude that efficient routes are established under all three circumstances.

Chapter 5

Experiments and results

To evaluate the performance of RT-AODV we want do to a large scale experiment with a long duration. This is infeasible due to the unavailability of a large testbed and the amount of time this would require. Therefore, we have decided to extend our proof of concept, as discussed in Chapter 4, by larger scale, one week simulation-time experiments.

To provide an extensive evaluation of RT-AODV, we have set up a realistic simulation environment and ran simulations of SOWNet's art monitoring application [1]. We compare RT-AODV to two protocols: the Collection Tree Protocol and NST-AODV (without acknowledgements and retransmissions). The powerful lab workstations allowed us to run multiple simulations concurrently, saving a lot of time.

We only evaluate the reliable data-collection scenario. RT-AODV can also be used as a basis for a reliable reconfiguration mechanism that can address individual nodes.

This chapter discusses the experiments that were performed to show how RT-AODV performs as a mechanism for guaranteed delivery of status messages and alarm messages to the sink. First, we discuss the simulation environment in Section 5.1. We discuss the experiments and results in Section 5.2. In this section, RT-AODV is compared to the Collection Tree Protocol [5] and NST-AODV [10].

5.1 Simulation environment

For our simulations we have used TOSSIM [19]. TOSSIM is a discrete event simulator for TinyOS [20]; it can run TinyOS programs without modification. The simulator includes the complete radio stack of the *lnode* platform of SOWNet Technologies [1].

To simulate a real WSN deployment, we have taken SOWNet's art monitoring application. This application sends periodic status messages to the sink at a 5 minute interval and an alarm message when a sensor is triggered.

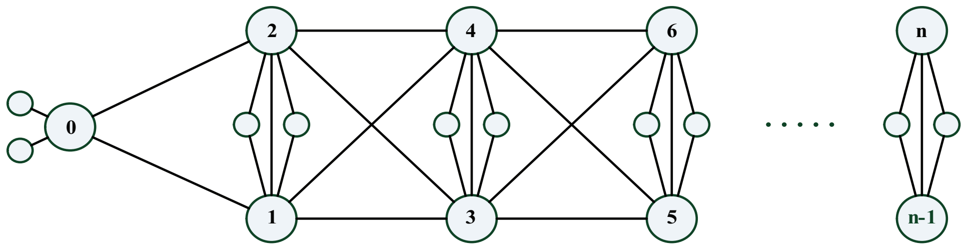


Figure 5.1: An example network-topology for an art monitoring system. Node 0 is the sink, the large circles are forwarding nodes, the small circles are sensor nodes. Each node is connected by a good link with each other node at a one hop distance, and by an unreliable link with nodes at a two hop distance. The unreliable links are not shown to keep the figure a bit clearer.

An example topology consists of 19 forwarding nodes, each connected to two sensing nodes, see Figure 5.1.

Each wireless network simulator requires a radio-channel model that determines for each message which nodes successfully received the message. Our simulator includes a Signal to Noise Ratio (SNR) based model and a link failure model. Both are discussed in more detail in the following paragraphs, followed by a paragraph on repeatability of experiments.

Radio model The simulator uses an SNR-based model to determine whether or not two nodes can exchange messages. A topology file defines the received signal strength in both directions between each pair of nodes in the network. Each radio has an SNR-threshold; if a signal exceeds the sum of all other signals by at least the SNR-threshold, it is decodable. A noise pattern that is loaded from a file is used to generate noise for each link.

Research has been done on the similarities between hardware experiments and simulations for SNR-based radio models [12]. The experiments are focused on MAC protocol performance. The results show very high similarities between simulations using on the SNR-based radio model and hardware experiments. One of the MAC protocols that is used for the evaluation is B-MAC. Because our simulation also uses B-MAC, our simulation results are expected to be realistic.

Link-failure model In addition to the SNR-based model, we have included a link-failure model to simulate environmental changes. Situations occur when nodes cannot communicate to each other for a short amount of time. This leads to bursty errors. We model those periods by randomly disabling an incoming or outgoing link for a period of two seconds. Data from deployments that are done by SOWNet show periods of bursty errors with a length of a few seconds.

Repeatability To be able to repeat an experiment several times, for example to compare how two protocols handle the same situation, we need to be able to repeat an experiment with the same series of events. We only do this for the link-failure model. Link failures occur less frequent than failing transmissions due to noise (determined by the SNR-model). Therefore the law of large numbers does not apply to link-failure scenarios. As link failures also have a much longer duration of two seconds, their influence on the results is much larger than the influence of noise.

To be able to repeat a link-failure scenario we seed the random number generator for each random event with a simulation-time dependent value and the *node id*. Time is divided in discrete blocks of two seconds. For each event that happens within a two second block, the random number generator is seeded with the start time of this block.

5.2 Experiments

In order to evaluate our work we have done simulations for various network sizes. Each run has a duration of one week simulation time. For each experiment, 10 runs are executed to obtain statistical significance. For the simulation experiments, we have evaluated RT-AODV under normal circumstances, i.e. there are no joining or leaving nodes and no network jamming.

In this section we evaluate the behaviour and performance of RT-AODV. The results are given in Table 5.1. We determine cost in terms of transmissions per end-to-end message. Overhead is calculated as the ratio of the number of control or beaconing transmissions to the number of data transmissions.

First we take a look at RT-AODV on its own; the behaviour during network setup and during stable operation are discussed. Then we compare RT-AODV to CTP and NST-AODV to see how it performs in contrast to these alternative solutions.

5.2.1 RT-AODV

We take a look at the behaviour of the protocol over one week. In the overview (Figure 5.2a) we see that most messages are delivered after one or two transmissions. A very small number of messages needed three or more transmissions. We conclude that RT-AODV determines routes with a good quality that require only a small number of transmissions.

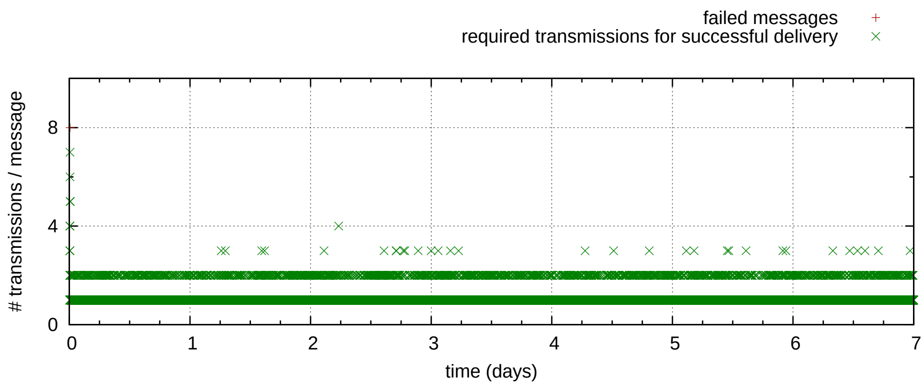
During network setup, the protocol has not yet established good routes. The first few messages of each forwarding node need more transmissions than is normal under stable operation. After that short period, the routes quickly stabilise; most messages are acknowledged after the first transmission. See Figure 5.2b. We conclude that RT-AODV has a low network setup time of a couple of minutes.

Table 5.1: Simulation results of experiments with three different protocols and different network sizes. Control cost and control overhead of CTP are not mentioned because CTP has only one packet for both control and beaconing purposes.

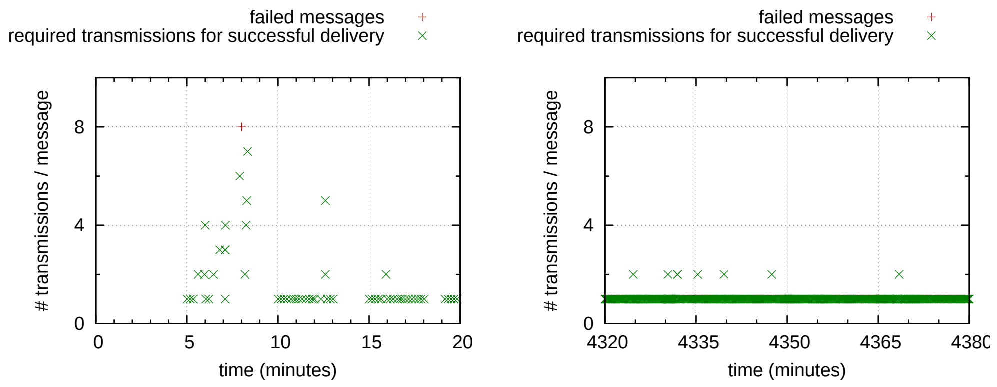
	RT-AODV				**CTP**			**NST-AODV**		
network size										
number of nodes	19	31	43	43-hb[a]	19	31	43	19	31	43
network diameter (hops)	9	15	21	21	9	15	21	9	15	21
delivery ratio (%)										
destination reached	100.00	99.40	97.55	99.96	80.32	59.88	46.91	99.46	99.30	99.13
standard deviation	(0.00)	(1.02)	(0.24)	(0.05)	(0.47)	(0.22)	(0.20)	(0.02)	(0.02)	(0.02)
acknowledged	99.97	97.06	89.39	99.84	-	-	-	-	-	-
standard deviation	(0.08)	(5.46)	(6.40)	(0.17)	-	-	-	-	-	-
cost (transmissions/end-to-end message)										
data message	16.95	33.43	54.12	16.55	-	-	-	-	-	-
forward message	8.48	16.72	27.06	16.55	10.76	21.85	34.54	10.08	14.90	23.03
acknowledgement	8.48	16.72	27.06	33.09	-	-	-	-	-	-
control	0.07	1.78	8.33	1.99	-	-	-	0.03	0.04	0.08
beaconing	0.29	0.31	0.33	0.30	0.58	1.41	2.40	0.38	0.34	0.38
total	17.32	35.52	63.58	35.38	11.33	23.26	36.94	10.48	15.26	23.48
total/network diameter	1.92	2.37	3.03	1.68	1.26	1.41	1.75	1.16	1.01	1.12
overhead (%)										
acknowledgements	100	100	100	100	-	-	-	-	-	-
control	0.44	5.08	15.79	5.91	-	-	-	0.24	0.28	0.33
beaconing	1.74	0.92	1.16	1.16	5.35	6.95	6.33	3.72	2.27	1.64
total	2.17	5.57	16.94	7.07	5.35	6.95	6.33	3.97	2.55	1.97

[a]Higher bandwidth: 125kbaud

During stable operation most messages are delivered and acknowledged after the first transmission. We can see only a small number of messages that requires a single retransmission; 98.9 % of the messages is acknowledged after the first transmission, 1.1 % of the messages after the second. See Figures 5.2a and 5.2c. RT-AODV operates very well under stable conditions. The routing and link-quality information is accurate and leads to stable routes.



(a) Overview of the 7 day experiment. 98.9 % of all messages was successfully acknowledged after one transmission. 1.1 % needed two transmissions. Less than 0.01 % of the messages was transmitted more than twice before an acknowledgement had been received.



(b) Initial 20 minutes. Only one message failed during the setup phase.

(c) Stable one hour period, starting at day 3.

Figure 5.2: 7 day simulation run.

5.2.2 Comparison to other protocols

We compare RT-AODV to two other protocols on delivery ratio and delivery cost. The first protocol is the Collection Tree Protocol; this is the default collection protocol that comes with TinyOS 2. We compare RT-AODV to CTP to show how it performs in contrast to the de facto solution for collection. CTP is discussed in more detail in Chapter 2. The second protocol is plain NST-AODV, i.e. RT-AODV without end-to-end acknowledgements and without end-to-end retransmissions.

Scalability We can see a fairly large performance drop in the results of RT-AODV for the 43 node network. In all experiments, the nodes near the sink have a very good delivery ratio (95 - 100 %). The 10 to 15 nodes that are furthest away from the sink show a large drop in message delivery (30 - 80 %). We have looked into various possible causes for the sudden performance drop for this specific group of nodes: route discovery problems, routing loops, and clustering of message injection over the forwarding nodes and in time, and bandwidth shortage.

If suddenly multiple nodes have problems in delivering their messages, a lot of retransmissions and control messages are send. Due to overhearing, only four nodes can send concurrently in the 43 node network. The combination of these causes a bandwidth shortage during which very little messages are successfully delivered. An experiment with more bandwidth shows that RT-AODV can also perform well on larger deployments.

Delivery ratio To compare the three protocols on delivery ratio we look at the ratio of acknowledged messages for RT-AODV. However, a message that is not acknowledged, did not per se fail to reach its destination. Therefore, we also look at ratio of delivered messages for RT-AODV. The results are given in Figure 5.3.

RT-AODV shows very good results for the network of 19 nodes. Very close to a 100 % of all messages is acknowledged. For the networks of 31 and 43 nodes, we can see a fairly large drop in performance, especially for 43 nodes. This has been discussed in the previous paragraph. The experiment with an increased bandwidth of 125 kbaud shows a delivery ratio of 99.84 %.

CTP shows very weak performance for all network sizes. Starting with only 80 % delivery for the smallest network, and further decreasing to 60 % and 47 % for the networks of 31 and 43 nodes respectively.

NST-AODV shows good results in terms of delivery ratio. For all network sizes, more than 99 % of the messages has successfully reached the sink. This is somewhat better than the delivery ratio of RT-AODV, which drops to 98 % for the largest network. For the network of 19 nodes, RT-AODV acknowledges more messages (99.97 %) than what NST-AODV delivers (99.46 %). For 31 nodes, the delivery ratios of both protocols are almost equal (RT-AODV: 99.40 %, NST-AODV: 99.30 %).

The difference in performance between RT-AODV and NST-AODV for 43 nodes can be due to higher network utilisation, or the maintenance of more routes. More route maintenance could introduce more routing loops. The problems that occur with RT-AODV, as discussed in the previous paragraph on scalability, do not seem to occur with NST-AODV.

The strength of RT-AODV is guaranteed delivery (acknowledgements), and not pure delivery ratio.

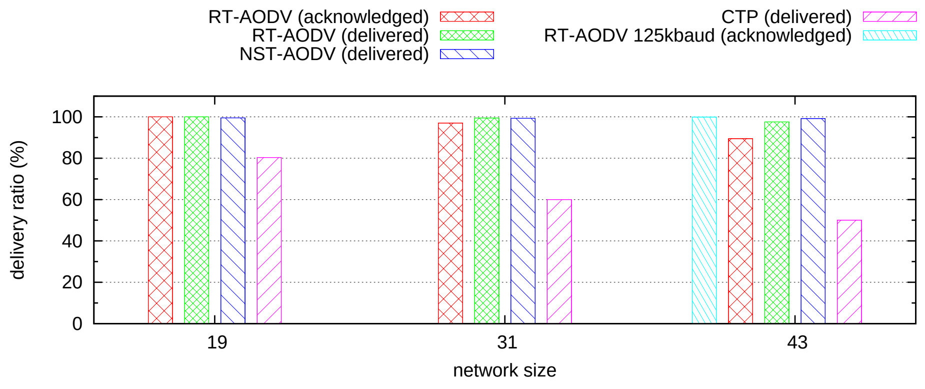


Figure 5.3: Delivery ratio of RT-AODV, CTP, and NST-AODV for network sizes 19, 31, and 43. For RT-AODV both the ratio of messages that are actually delivered to the destination, and the ratio of messages that are acknowledged, are plotted. For CTP and NST-AODV only the ratio of delivered messages is plotted.

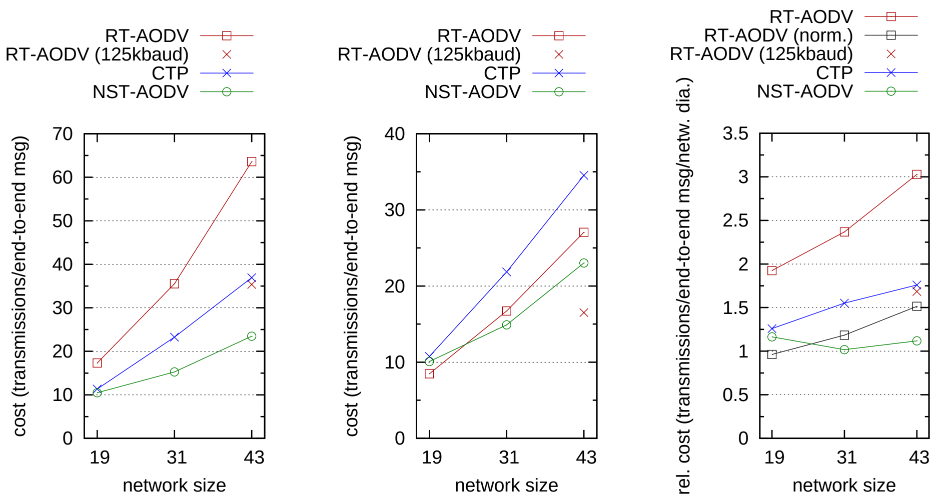
Delivery cost To compare the cost for the three protocols we look at the total cost in terms of transmitted messages per end-to-end message (Figure 5.4a), and the number of transmitted messages to only deliver the forward message (and not the acknowledgement). The latter evaluates how well the protocol determines the best available route; the fewer messages are required to reach the destination, the better the route is (Figure 5.4b).

The total cost for RT-AODV to deliver and acknowledge a message is rather high. The number of required messages varies from 2 times the network diameter for the smallest network to 3 times the network diameter for the largest network (Figure 5.4c). NST-AODV requires a minimum number of transmissions to deliver a message to its multi-hop destination; slightly more transmissions than the number of hops are required. This indicates that reliable and short routes are determined by the routing protocol. CTP requires a larger number of transmissions to deliver a message: 1.26 to 1.75 transmissions per hop.

When we compare the total delivery cost for RT-AODV to the other two protocols, we see that it has almost doubled (Figure 5.4a). Most of this increase is due to the acknowledgement that has to travel the complete route back to the source. For the larger networks, overhead also starts to count. Figure 5.4c shows cost relative to the network diameter; the acknowledgement is left out for *RT-AODV (norm.)*. The increase from 1 to 1.5 transmissions per hop is partly caused by increased overhead (Table 4.2).

The cost to deliver the message to its destination, not counting the acknowledgement, is very close to the minimum for NST-AODV. RT-AODV is very close. CTP is using much more transmissions than the other two protocols (Figure 5.4b).

If we compare the results from the RT-AODV experiment with a higher bandwidth of 125 kbaud, we see that delivery costs have severely decreased. Figure 5.4a and Figure 5.4c show that RT-AODV is very competitive to CTP.



(a) Total cost **(b)** Forward message **(c)** Relative cost

Figure 5.4: Delivery cost in terms of transmissions per end-to-end message for RT-AODV, CTP, and NST-AODV for network sizes 19, 31, and 43.

Chapter 6

Conclusions and future work

The motivation for this MSc thesis project is the need of SOWNet Technologies for a reliable transport protocol for wireless sensor networks. Part of the requirement is that the protocol must be able to deliver an individual message (i.e. no bulk transport: a message that is not part of a stream or block). Previous work does not include end-to-end acknowledgements; reliable transport for WSNs mainly consists of negative acknowledgement mechanisms, which are only suited for bulk transport.

Our solution, called RT-AODV, treats each message on its own. The reactive any-to-any routing protocol NST-AODV is used to deliver messages to their multi-hop destination. If a message is delivered to its destination, an end-to-end acknowledgement is sent back to the source. If it takes too long for an acknowledgement to be received by the source, a timeout is fired and the message is retransmitted. If the maximum number of retransmits has been reached, delivery has failed. The goal of our solution is twofold: provide guaranteed delivery for an individual message, and provide the means to do reliable reconfiguration for single nodes.

We have made improvements to NST-AODV to enhance its robustness, and made additions to include a statistical link-estimator, loop detection, and route monitoring. On top of NST-AODV we have implemented an end-to-end acknowledgement mechanism with timeouts and retransmissions.

To evaluate RT-AODV we have done a proof of concept implementation on the testbed of the Embedded Software group of Delft University of Technology. To be able to evaluate our work on a larger scale, with a longer duration, and with respect to other protocols, we have done TOSSIM simulations. We have compared RT-AODV to NST-AODV and the Collection Tree Protocol. The remainder of this section draws conclusions on the used techniques and the outcome of the performed experiments.

Conclusions We have used an any-to-any routing paradigm to deliver messages from all nodes to the sink (i.e. data collection). The sink requires a route to the source of each message to deliver an acknowledgement. To be able to reach each node in the network, a routing table entry is required for each forwarding node in the network. This requires a fair amount of state for each node. Due to advances in microcontrollers, more and more RAM becomes available, and maintaining more state is less of a problem. If larger networks are required, for which the amount of state cannot fit in RAM, we can look for hybrid solutions of source routing and hop-by-hop routing, as is discussed below.

The use of end-to-end acknowledgements increases the delivery cost, in terms of required messages, by a factor of two. This is a direct implication of end-to-end acknowledgements; we need to send a data message to the destination, and an acknowledgement message back to the source. This makes end-to-end acknowledgements an expensive approach to reliable transport.

The overall performance of RT-AODV is good. It operates very stable under typical conditions. If the complete network, or a part of the network, cannot transmit messages due to radio jamming, this leads to failed messages. This is something that any protocol would suffer from. After the network jamming is over, the network quickly stabilises.

Network setup time is low. This is the period when nodes start to transmit their first messages, and no routes are available yet. Some of the first messages fail and require more retransmissions. After a few minutes, when all routes are established, the network becomes stable and messages are delivered and acknowledged after the first or second transmission.

Compared to CTP, RT-AODV provides a much better delivery ratio at a significant cost increase. As noted before, this is a direct consequence of using end-to-end acknowledgements. Compared to NST-AODV, RT-AODV delivers the same ratio of messages to the destination. With respect to NST-AODV and CTP, the acknowledgement mechanism is the major strength of RT-AODV. If an acknowledgement has been received, the source can tell for sure that the message has been delivered to the destination.

We conclude that any-to-any routing and end-to-end acknowledgements are a good approach for reliable transport in low bandwidth and low resource networks like WSNs.

Future work Future research in reliable transport for wireless sensor networks could include a hybrid solution of source routing and hop-by-hop routing. An approach to reduce the overhead of end-to-end acknowledgements can start off from a single-hop acknowledgement protocol that is combined with a negative-acknowledgement flooding mechanism.

Routing To decrease the routing table size that is required to address all nodes in the network, we could introduce source routing for the first few hops of each route that originates from the sink.

For this approach, the sink maintains the first few hops for each destination. This might require specialised hardware, as more memory is required to store those routes. If the first few hops are maintained on the sink node the close neighbours of the sink do not need to maintain routes to a large part of the network. This decreases the memory required for the routing table. The last few hops can be included in the message header, and updated at each transmission. All nodes maintain a route to the sink; it only costs a single routing table entry for the complete network. When a message is received by the sink, it records the last hops as the start of the route to the source of the message. When the message reaches the end of the route that is provided in the header, the current node can send over the its known route to the destination, or start a route request if no route is available.

This mechanism only works if the *connection* is initialised by a node that wants to transmit to the sink. When the sink initialises a connection to a sensor node for which it has not received any messages, it does not have a route to that destination. Some sort of route discovery is required to reach a node that does have a valid route. We can use a route discovery mechanism like it is implemented in NST-AODV. If the destination is reached, a route reply is send back to the sink over the same route as the route request, and records its last few hops. Upon reception of the route reply by the sink, it records the start of the route, and a valid route to the destination has been created.

This approach is a trade-off between an increase in routing table size (hop-by-hop routing) and an increase in message size (source routing). An advantage is that only a single node (the sink) requires additional memory (flash or RAM) to store the start of each route.

Reliable transport The overhead that is introduced by end-to-end acknowledgements is quite high; it takes up almost half of the delivery cost per message. A reliability protocol that only compromises slightly on guaranteed delivery, but has a heavily reduced overhead, is a good subject for future research.

We can leave out end-to-end acknowledgements if there is only a small number of messages that is not delivered with the first transmission. Then only a small number of failed deliveries remains, for which a failure message has to be delivered back to the source. This message must be delivered with very high probability to provide reliable transport. Failures can be detected at forwarding nodes when they do not have a route to forward the message to the destination. The number of failure messages will be relatively small; the increase in overall delivery cost due to an expensive delivery method will be small. One can think of a reliable form of flooding or gossipping to deliver the message. Direct neighbours can require an acknowledgement from the source node.

If a failure message reaches the source, it can retransmit the message with a request for an end-to-end acknowledgement. If a node notices network of delivery problems in its vicinity, there might be a higher probability for its own messages to fail as well. Under those conditions, a node can request to receive an end-to-end acknowledgement for a message.

This approach can heavily reduce the overhead that is introduced by end-to-end acknowledgements.

Bibliography

[1] SOWNet Technologies. http://www.sownet.nl, september 2009.

[2] P. Buonadonna, J Hill, and D Culler. Active message communication for tiny networked sensors, 2001.

[3] T. Clausen and P. Jacquet. Optimized link state routing protocol (olsr), 2003.

[4] D.S.J. De Couto, D. Aguayo, J. Bicket, and R. Morris. a high-throughput path metric for multi-hop wireless routing. *Wireless Networks*, 11(4):419–434, July 2005.

[5] R. Fonseca, O. Gnawali, K. Jamieson, S. Kim, P. Levis, and A. Woo. TEP 123: The Collection Tree Protocol, August 2006. `http://www.tinyos.net/tinyos-2.x/doc/html/tep123.html`.

[6] R. Fonseca, O. Gnawali, K. Jamieson, and P. Levis. Four bit wireless link estimation. In *Proceedings of the Sixth Workshop on Hot Topics in Networks (HotNets VI)*, 2007.

[7] D. Gay, P. Levis, R. von Behren, M. Welsh, E. Brewer, and D. Culler. The nesc language: A holistic approach to networked embedded systems. In *PLDI '03: Proceedings of the ACM SIGPLAN 2003 conference on Programming language design and implementation*, pages 1–11, New York, NY, USA, 2003. ACM.

[8] O. Gnawali. Tep 124: The link estimation exchange protocol (LEEP), February 2006. `http://www.tinyos.net/tinyos-2.x/doc/html/tep124.html`.

[9] O. Gnawali, R. Fonseca, K. Jamieson, D. Moss, and P. Levis. Collection Tree Protocol. In *Proceedings of the 7th ACM Conference on Embedded Networked Sensor Systems (SenSys'09)*, November 2009.

[10] C. Gomez, P. Salvatella, O. Alonso, and J. Paradells. Adapting aodv for ieee 802.15.4 mesh sensor networks: Theoretical discussion and performance evaluation in a real environment. In *WOWMOM '06: Proceedings of the 2006 International Symposium on on World of Wireless, Mobile and Multimedia Networks*, pages 159–170, Washington, DC, USA, 2006. IEEE Computer Society.

[11] Z.J. Haas and M.R. Pearlman. *ZRP: a hybrid framework for routing in Ad Hoc networks*, pages 221–253. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 2001.

[12] G.P. Halkes and K.G. Langendoen. Experimental evaluation of simulation abstractions for wireless sensor network mac protocols. In *14th IEEE International Workshop on Computer Aided Modeling and Design of Communication Links and Networks (IEEE CAMAD '09)*, 2009.

[13] I. Haratcherev, G. Halkes, T. Parker, O. Visser, and K. Langendoen. PowerBench: A Scalable Testbed Infrastructure for Benchmarking Power Consumption. In *Int. Workshop on Sensor Network Engineering (IWSNE)*, 2008.

[14] T. He, S. Krishnamurthy, J.A. Stankovic, T. Abdelzaher, L. Luo, R. Stoleru, T. Yan, L. Gu, J. Hui, and B. Krogh. Energy-efficient surveillance system using wireless sensor networks. In *MobiSys '04: Proceedings of the 2nd international*

conference on Mobile systems, applications, and services, pages 270–283, New York, NY, USA, 2004. ACM.

[15] Texas Instruments. CC2420 datasheet, november 2009.

[16] D.B. Johnson, D.A. Maltz, and J. Broch. *DSR: the dynamic source routing protocol for multihop wireless ad hoc networks*, pages 139–172. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 2001.

[17] J. M. Kahn, R. H. Katz, and K. S. J. Pister. Next century challenges: mobile networking for "smart dust". In *MobiCom '99: Proceedings of the 5th annual ACM/IEEE international conference on Mobile computing and networking*, pages 271–278, New York, NY, USA, 1999. ACM.

[18] S. Kim, R. Fonseca, P. Dutta, A. Tavakoli, D. Culler, P. Levis, S. Shenker, and I. Stoica. Flush: a reliable bulk transport protocol for multihop wireless networks. In *SenSys '07: Proceedings of the 5th international conference on Embedded networked sensor systems*, pages 351–365, New York, NY, USA, 2007. ACM.

[19] P. Levis. Tossim system description. http://www.tinyos.net/nest/doc/tossim.pdf, January 2002.

[20] P. Levis, S. Madden, J. Polastre, R. Szewczyk, K. Whitehouse, A. Woo, D. Gay, J. Hill, M. Welsh, E. Brewer, and D. Culler. Tinyos: An operating system for sensor networks. pages 115–148. 2005.

[21] A. Mainwaring, D. Culler, J. Polastre, R. Szewczyk, and J. Anderson. Wireless sensor networks for habitat monitoring. In *WSNA '02: Proceedings of the 1st ACM international workshop on Wireless sensor networks and applications*, pages 88–97, New York, NY, USA, 2002. ACM.

[22] J. Paek and R. Govindan. Rcrt: rate-controlled reliable transport for wireless sensor networks. In *SenSys '07: Proceedings of the 5th international conference on Embedded networked sensor systems*, pages 305–319, New York, NY, USA, 2007. ACM.

[23] C.E. Perkins, E. Belding-Royer, and S. Das. Ad hoc on-demand distance vector (AODV) routing. RFC 3561, July 2003. Network Working Group.

[24] C.E. Perkins and P. Bhagwat. Highly dynamic destination-sequenced distance-vector routing (dsdv) for mobile computers. *SIGCOMM Comput. Commun. Rev.*, 24(4):234–244, 1994.

[25] R. Simón Carbajo, M. Huggard, and C. McGoldrick. An end-to-end routing protocol for peer-to-peer communication in wireless sensor networks. In *MiNEMA '08: Proceedings of the 6th workshop on Middleware for network eccentric and mobile applications*, pages 5–9, New York, NY, USA, 2008. ACM.

[26] K. Srinivasan, P. Dutta, A. Tavakoli, and P. Levis. Some implications of low power wireless to IP networking. In *The Fifth Workshop on Hot Topics in Networks (HotNets-V)*, November 2006.

[27] C. Wan, A.T. Campbell, and L. Krishnamurthy. Psfq: a reliable transport protocol for wireless sensor networks. In *WSNA '02: Proceedings of the 1st ACM international workshop on Wireless sensor networks and applications*, pages 1–11, New York, NY, USA, 2002. ACM.

[28] G. Werner-Allen, J. Johnson, M. Ruiz, J. Lees, and M. Welsh. Monitoring volcanic eruptions with a wireless sensor network. *Wireless Sensor Networks, 2005. Proceeedings of the Second European Workshop on*, 1:108–120, Jan.-2 Feb. 2005.