Implementing a Cooperation and Accounting Strategy for Multi-hop Cellular Networks

Diplomarbeit
der Philosophisch-naturwissenschaftlichen Fakultät
der Universität Bern

vorgelegt von

Thomas Staub
2004

Leiter der Arbeit:
Professor Dr. Torsten Braun
Institut für Informatik und angewandte Mathematik

Contents

List of Figures

List of Tables

Listings

Acknowledgment

First of all, I would like to dedicate some words to the people who contributed to the success of my thesis.
I owe many thanks to Professor Dr. Torsten Braun for supervising my thesis. Furthermore, I would like to express my gratitude to Attila Weyland. He supported me during the whole diploma work, gave me many valuable hints and recommendations, and sacrificed a lot of his spare-time in order to proof-read my thesis several times. Many thanks go to Dr. Florian Baumgartner and Matthias Scheidegger for providing technical support. Moreover, I want to thank Niklaus Steiner and Vivian Kilchherr for proof-reading this thesis. Finally, I would like to thank my family and my girlfriend for their support.

Chapter 1

Introduction

Todays wireless networks consist of a number of access points deployed by a provider in limited areas where a certain amount of customers are expected, e.g. at railway stations or at airports. Potential customers outside the area covered by the provider's access points can not participate in the network. Only the customers in the restricted area of the access points can provide revenue for the provider. The only solution is to deploy more access points which can be limited by location properties in addition to financial issues.

A promising concept to cover bigger areas with wireless networks is a multi-hop cellular network. In this network the single hop limit to the access point is removed. The customers act as packet forwarders and a gateway offers the Internet connectivity. The size of the covered area increases without deploying new access points. The advantages of a mobile ad-hoc network are combined with the existing network infrastructure. But this concept includes also some drawbacks of a mobile ad-hoc network. Routing information has to be maintained accurate, the customers have to be protected from attacks that are possible because of the open architecture, as well as the cooperation among the customers has to be ensured.

The willingness to cooperate in such a network becomes a challenge. The customers tend to give priority to their self-generated packets over packets of other customers, although they can have a common interest in Internet connectivity. This problem becomes more critical when energy is regarded as a precious and limited good. No-one would forward packets for the benefit of others if as consequence he has no energy left for the transmission of his own packets. This natural selfishness of the customers prevents such a network to be kept alive.

The CASHnet [16, 17, 18] cooperation scheme addresses the shown issues in a decentralized manner. It accepts that the individual customer plays an important role in such a network and his participation must be encouraged. It stimulates the cooperation by making it a rewarding alternative to the selfishness. The customers have to pay for their self-generated packets with a certain amount of Traffic Credits. The customers that forward packets for the benefit of others are rewarded and can save money by not having to buy Traffic Credits.

For this thesis an implementation of the promising CASHnet scheme is made in a network simulator to verify its feasibility. Furthermore, another cooperation scheme called Nuglet [19] is implemented to make a comparison with the CASHnet scheme.

This thesis is structured as follows. The chapter 2 provides an overview of the different types of wireless networks. In the chapter 3 the different routing concepts in a mobile ad-hoc

network are illustrated and a closer look is taken to the used routing protocol AODV (Ad-hoc On-Demand Distance Vector) and an enhancement of it. In the chapter 4 the cooperation in mobile ad-hoc networks, different cooperation concepts, the CASHnet and the Nuglet scheme are discussed. The implementations of CASHnet and Nuglet in the network simulator are treated in chapter 5. The chapter 6 shows the simulation scenarios and their evaluations. Finally, the chapter 7 presents the made conclusions and an outlook to future work.

Chapter 2

Wireless Networks

In the last years, the popularity of wireless networks increased. Wireless networks offer many advantages in the form of availability, mobility and adaptability to the users. They are built in environments where the installation of wires is not possible or not wished. They require less infrastructure than wired networks and can be set up faster, e.g. in an emergency mission. Furthermore, they provide mobility to the users by freeing them of dangling cables. There exist three types of wireless networks: *mobile ad-hoc networks*, *cellular networks* and *multi-hop cellular networks*. They are discussed in this chapter.

2.1 Mobile Ad-hoc Networks

A mobile ad-hoc network (MANET) consists of a collection of mobile nodes which have the possibility to connect to a wireless medium and form a dynamic network with wireless links. Since the nodes are mobile, the links between them are not permanent. The network topology may change rapidly and unpredictably in time. New nodes can join the network, and other nodes may leave the network.

The expected size of a MANET is larger than the transmission range of the nodes, because of this fact it is necessary to route the traffic through a multi-hop path for giving the nodes the ability to communicate with each other. There exist neither fixed routers nor fixed locations for the routers nor centralized administration. The lack of any fixed infrastructure is compensated by the routing ability of every mobile node. They all act as mobile routers and for this they need the capability to discover and maintain routes to every node in the network and to route the packets accordingly.

Possible applications of MANET are in scenarios with little or no communication infrastructure: emergency relief, military operations, or situation where people wish to simply share information, e.g. at a conference.

2.2 Cellular Networks

Cellular networks or infrastructure networks are based on a wired back-bone which connects the base-stations. The base-station nodes have at least one network interface for the wired network

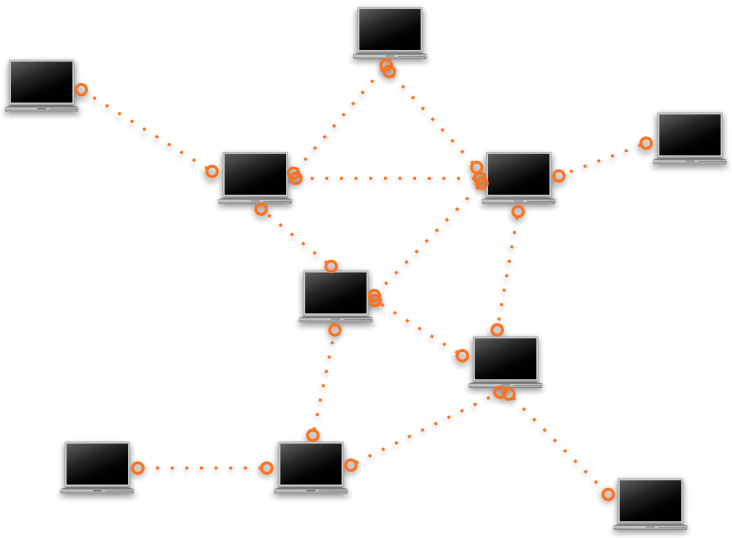


Figure 2.1: Mobile ad-hoc network

and one or more wireless network interfaces to provide communication to the mobile nodes. The communication of the mobile node is only possible over a one-hop link to base-station. Direct links between nodes or multi-hop links to the base-station are not possible.

The size of a cellular network is limited by the transmission range of the base-stations. If the

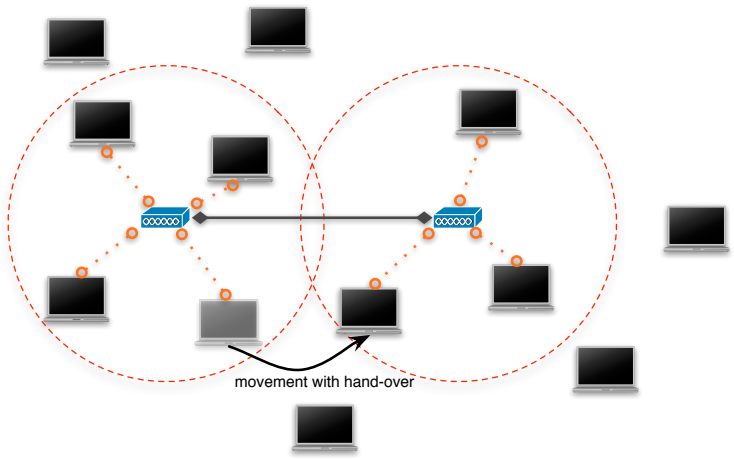


Figure 2.2: Cellular network

node is out of the transmission range of the base-stations, no communication is possible. Inside the area covered by the base-stations it may move without losing connection and if it leaves the transmission range of the current base-station, a hand-over to a another base-station will let the node communicate seamlessly.

2.3 Multi-hop Cellular Networks

In multi-hop cellular networks the two concepts described before are combined. On the one hand there is a cellular network, on the other hand there are mobile nodes with additional routing facilities. With this approach it is possible to have multiple hops between a mobile node and a base-station. The idea is to benefit from existing infrastructure and to gain more efficiency out of it, to cover wider areas with less fixed antennas and base-stations and to reduce power consumption due to shorter hop distances. In [2] some benefits of enhancing cellular network with ad-hoc technologies are shown.

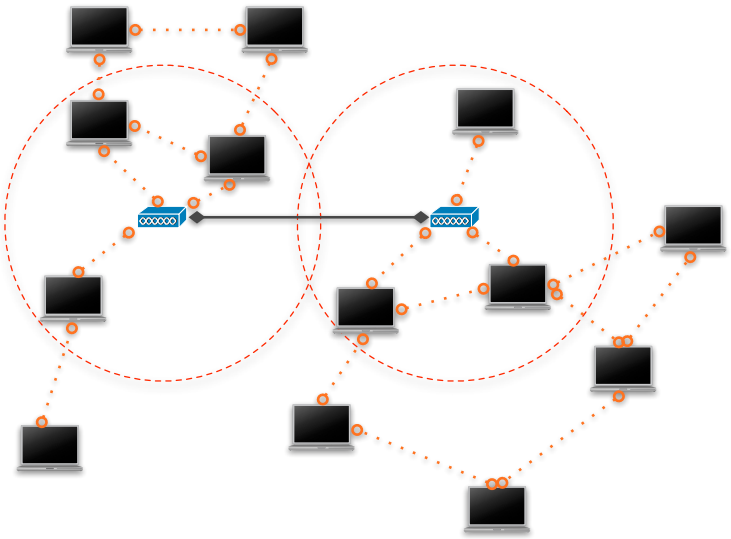


Figure 2.3: Multi-hop cellular network

Chapter 3

Routing in Mobile Ad-hoc Networks

In this chapter different concepts of routing in MANET are shown. Further the mobile ad-hoc routing protocol AODV (Ad-hoc On-Demand Distance Vector) is discussed in more detail because it is used in the implementation of CASHnet and Nuglet.

3.1 Concepts

A routing protocol for a mobile ad-hoc environment is in urgent need of the following capabilities:

- Loop-freeness
- Multi-hop paths
- Self-starting
- Dynamic maintenance of the network topology
- Fast convergence
- Minimal Routing overhead
- Economical consumption of resources, e.g. memory and bandwidth
- Minimized and kept local effect of link breakage
- Scalability with large numbers of nodes

Different concepts for mobile ad-hoc routing are established in order to reach these capabilities. There are different interdependences between the wished capabilities. Because of this, there exists no concept that is optimal in all aspects. Each approach has to make a compromise on the different capabilities.

3.1.1 Position Based Protocols

This concept makes use of location information [3]. The routing can be based on the location information either to flood route requests or to forward the data packets. The basic components of a position based routing are:

- positioning service to determine the physical position of the node, e.g. GPS
- location service to determine the position of the destination, e.g. DREAM [4], Quorum based
- forwarding strategy, i.e. selection of the next node

3.1.2 Topology Based or Non-Position Based Protocols

The topology based protocols do not make use of additional location information. They utilize network topology information to make a routing decision.

Proactive Protocols

In proactive or table-driven protocols, the nodes in the network maintain a table of routes to every destination. They periodically exchange messages to keep the routing table up-to-date. At all times the routes to all destinations are ready to use which keeps the delays to send data packets. The maintenance of routes to all destinations, even if they are not used, consumes a lot of bandwidth and network resources. It can even end in increasing delays because of queues filled up with control packets and more packet collisions due to more network traffic. As a result proactive protocols do not scale in the frequency of topology change. Therefore they are only appropriate for low mobility networks.

Representatives of proactive protocols are DSDV (Destination-Sequenced Distance Vector Routing) [6] and OLSR (Optimized Link State Routing) [7, 8].
DSDV is a distance vector protocol which uses the Bellmann-Ford algorithm. A sequence number is added to guarantee loop-freedom by distinguishing stale routes from new ones.
OLSR is a proactive link state routing protocol. Its neighbor sensing is based on periodic exchange of $HELLO$ messages. It reduces the flooding of control traffic by using the concept of multi-point relays and computes the routes with the shortest-path algorithm.

Reactive Protocols

Reactive (or on-demand) protocols acquire only routing information upon request. They are designed to overcome the wasted effort in maintaining unused routes. Routes are searched on-demand. When a node requires a new route to a destination, it starts a route discovery process. This process ends once that a valid route is found or all possible routes are checked. The nodes are not forced to maintain unused routes, but on the other hand the latency for sending data packets will considerably increase. A long delay before data transmission can arise because the transmission has to wait until a valid route to the destination is acquired. As reactive routing

protocols flood the network to discover the wished route, they are not optimal in terms of bandwidth utilization, but scale well in highly dynamic networks. Thus this strategy is suitable for high mobility networks.

Exponents of this strategy are e.g. TORA (Temporally-Ordered Routing Algorithm) [9], DSR (Dynamic Source Routing) [10] and AODV (Ad-Hoc On-Demand Distance Vector Routing) [11, 12].
TORA is built on the concept of link reversal. The main design concept is to reduce the control messages to few nodes near the topology change by using a destination based acyclic graph.
DSR is based on source routing. A sender has to get the complete sequence of nodes to the destination and includes this list of intermediate nodes in the packet header before sending the packet. A closer look to AODV is taken in the following section.

3.2 Examples

3.2.1 AODV

AODV [11, 12] is a reactive mobile ad-hoc routing protocol. It joins the mechanisms of DSDV and DSR. The periodic beacons, hop-by-hop routing and the sequence numbers of DSDV and the pure on-demand mechanism of *Route Discovery* and *Route Maintenance* of DSR are combined.

As an important feature AODV uses a destination sequence number for each route entry. This destination sequence number is generated by the destination node and is sent to the requesting node. This trivially insures loop-freedom by simply selecting the route with the highest sequence number as the actual one.

Message Format

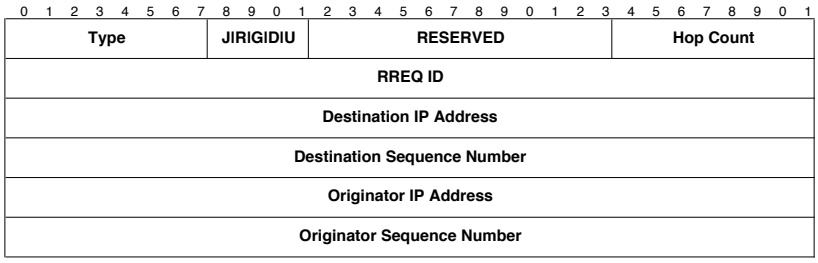


Figure 3.1: AODV: Route Request ($RREQ$)

AODV has four types of messages: Route Requests ($RREQ$), Route Replies ($RREP$), Route Errors ($RERR$), and Route Replies Acknowledgment ($RREP - ACK$). The different messages are shown in Fig. 3.1 - 3.4. All these messages are received via UDP using normal IP header processing. AODV uses the IP limited broadcast address (255.255.255.255) to broadcast messages.

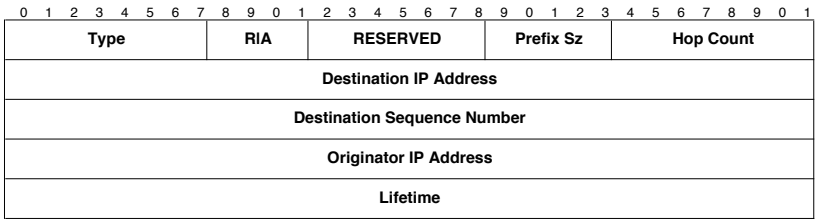


Figure 3.2: AODV: Route Reply ($RREP$)

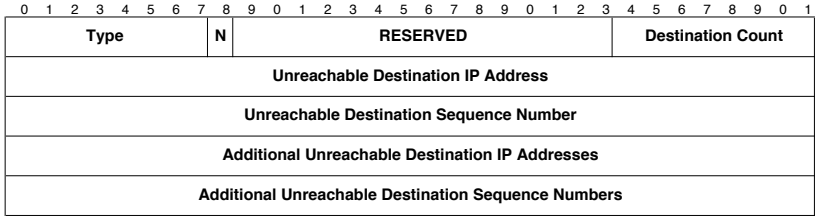


Figure 3.3: AODV: Route Error ($RERR$)

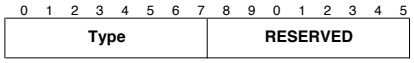


Figure 3.4: AODV: Route Reply Acknowledgment ($RREP - ACK$)

$Type$ 1 for $RREQ$
2 for $RREP$
3 for $RERR$
4 for $RREP - ACK$

J Join Flag; reserved for multicast.

R Repair Flag; reserved for multicast.

G Gratuitous $RREP$ flag; indicates whether a gratuitous $RREP$ should be unicast to the destination node.

D Destination only flag; only the destination node may answer to this $RREQ$, no intermediate node is allowed of answering with a $RREP$.

U Unknown sequence number.

A Acknowledgment required; used, if there is a danger of unidirectional links. It causes the receiver of the $RREP$ message to send back a $RREP - ACK$ message. The reception of such an acknowledgment provides assurance that the link is currently bidirectional.

N No delete flag; set if upstream nodes should not delete the route, although a node has performed a local repair of a link.

Sequence Number and Routing Table Management

It is crucial for AODV to properly handle the sequence numbers. A node has to update its own sequence number in two cases:

- Before starting a route discovery process, the node has to increment its own sequence number.
- A destination node has to update its own sequence number to the maximum of its current sequence number and the destination sequence number in $RREQ$ packet immediately before transmitting the $RREP$ packet.

The sequence numbers in the routing table entries may be changed by the node only in the following circumstances:

- Offer of a new route to itself, if it is the destination node.
- Reception of an AODV message with new information about the sequence number for a destination.
- Expiration of path or path breaks.

When a node receives an AODV control message, either to create or to update a route for a particular destination, it searches its routing table for an entry to the destination. If there is no route entry, it creates a new one with the sequence number contained in the control packet, or else the sequence number is set invalid. Otherwise, the node compares the existing entry with the new information and updates it if either

- the new sequence number is higher than in the routing table entry,
- the sequence numbers are equal and the new hop count plus one is smaller than in the existing route, or
- the sequence number is unknown.

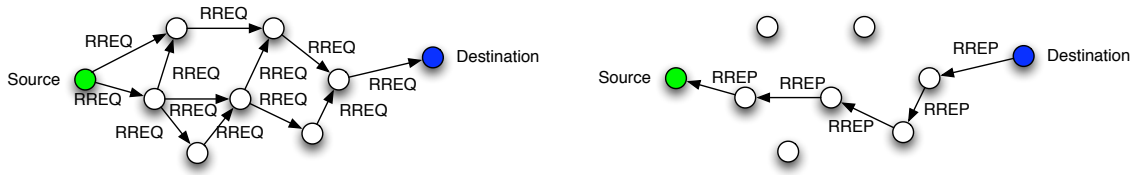
Besides the destination sequence numbers, the routing entry for each valid route contains a precursor list. This list contains all precursor of the node which are able to forward packets on this route. All neighboring nodes to which a $RREP$ was generated or forwarded are included in this list. In the event of a next hop link breakage, notifications are sent to those nodes.
The routing table entries of AODV consist of the following entries:

- Destination IP Address
- Destination Sequence Number
- Valid Destination Sequence Number flag
- Routing and state flags
- Network Interface

- Hop Count (distance in hops to reach the destination)
- Next Hop
- List of Precursors (as mentioned above)
- Lifetime (deletion time of the route entry)

Route Discovery

If a valid route exists between two communication peers, AODV takes no action. When a new route is needed, the *Route Discovery* mechanism is started. The source node has to send a $RREQ$ message. The sequence number field in the $RREQ$ is set to the last known destination sequence number or if not available the unknown sequence number field is set. The own sequence number is incremented and included in the Originator Sequence Number field of the message. The $RREQ$ ID field is incremented by one of the node's current $RREQ$ ID. The hop count is set to zero. The node buffers the $RREQ$ ID and the Originator IP address of $RREQ$ before broadcasting it. The source node waits now for a $RREP$ message. If it does not retrieve one within a certain time, it may broadcast another $RREQ$. If the maximum number of retries has been reached, all data packets for this destination are dropped and a destination unreachable message is delivered to their originators.



(a) Route Request ($RREQ$) **(b)** Route Reply ($RREP$)

Figure 3.5: AODV: Route Discovery

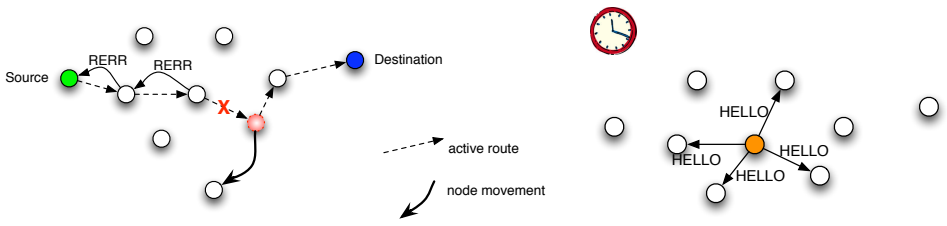
The intermediate node which is receiving the $RREQ$ checks if it has already received a message with the same Originator Address and $RREQ$ ID within a certain time. If a $RREQ$ has been already received, the newly received message is discarded. Otherwise, the node increments the hop count in the $RREQ$ and searches for a reverse route to the originator and creates or updates its route entry in the routing table. The destination sequence number of the reverse route in the routing table is set to the Originator Sequence Number if it is greater, the sequence number is set valid, the hop count is copied from the $RREQ$ and the next hop is changed. If the intermediate node has a fresh enough valid route to the destination, it unicasts a $RREP$, whose hop count is set to the hop distance of the current node to the destination, to the source node and discards the $RREQ$. Otherwise, it broadcasts the $RREQ$.

When the $RREQ$ reaches the destination node, the node sends a $RREP$ back towards the source of the $RREQ$ using the reverse route. The destination node increments its own sequence

number and puts it in the Destination Sequence Number field. The hop count is reset to zero. Each intermediate node forwarding the $RREP$ always increments the hop count. As soon as the source node retrieves the $RREP$, it is able to transmit the data packets to the destination.

Route Maintenance

Nodes which are part of an active route can deliver connectivity information by broadcasting $HELLO$ messages. A $HELLO$ message is a $RREP$ message with TTL = 1. By listening for packets from its neighbor nodes a node can determine the connectivity. If it receives neither $HELLO$ nor other messages from a certain node during a certain interval, it has to assume a link break. Local connectivity can also be surveyed by using link layer notification. The node



(a) Route Error ($RERR$) **(b)** Periodic $HELLO$ messages

Figure 3.6: AODV: Route Maintenance

sends a $RERR$ to all nodes in the precursor list of the concerned route, if it has detected a link break for the next hop of an active route, or if it gets data packets for a node for which it does not have an active route, or if it receives a $RERR$ from a neighbor for an active route.

3.2.2 AODV+

AODV+ [13] is an implementation of global connectivity for mobile ad hoc networks as presented in [14]. It is currently being implemented for the network simulator ns2 [15]. The used concepts are kept general and could be adapted to a real implementation.

The work is focused on interworking between a mobile ad-hoc network and the Internet. Gateways are needed to send packets outside the mobile ad-hoc network. These gateways are able to route messages in both networks. Therefore they support mobile ad-hoc routing and routing in the wired domain. [14] explains the operation of the mobile nodes and the gateways. The application of these concepts for a reactive routing protocol heads in adaptation of the route discovery messages ($RREP$, $RREQ$), so that it is possible to detect gateways. The $RREQ$ is extended by an additional flag which is called *Internet-Global Address Resolution Flag* or I-flag ($RREQ_I$). A set I-flag indicates that the source node requests global connectivity and wants to recover a gateway. The $RREP$ is also extended by the same additional I-flag. The new flag in the $RREP_I$ message indicates that the message contains information about a gateway.

In order to get access to the wired network, the node needs to learn the location and the address of a gateway. The route to the gateways is then used as default route to send packet outside the MANET. There are different ways to catch information about the gateways:

- the gateway broadcasts periodically messages (proactive gateway discovery)
- the mobile node requests a default route to a gateway by sending a $RREQ_I$ message (reactive gateway discovery)
- the gateways replies on received $RREQ$ with a $RREP_I$

The mobile node adds this information about the gateway to its routing table as the *default route* entry. If the mobile node has no route to a certain node, it broadcasts a $RREQ$. If it does not receive a $RREP$, the node is supposed to be outside the current mobile ad-hoc network. The packet is sent to this node by using the *default route* to the gateway.

Proactive Gateway Discovery

The gateway discovery is initiated by the gateways themselves. They broadcast periodically a *gateway advertisement* ($GWADV$) message to the whole mobile ad-hoc network. It is important to carefully choose the broadcasting interval to not flood the network unnecessarily.

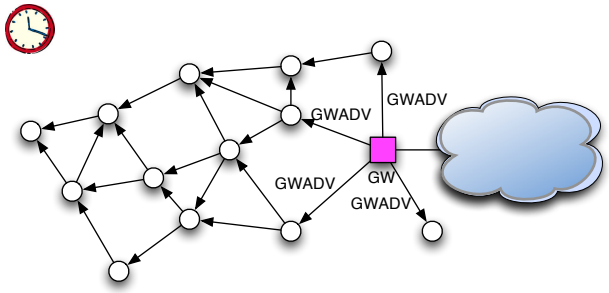
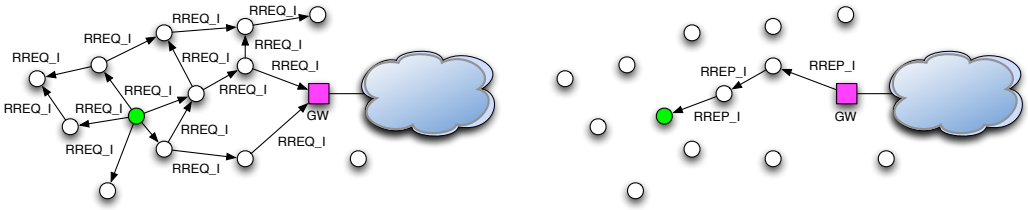


Figure 3.7: AODV+: Proactive Gateway Discovery

The mobile nodes receiving this $GWADV$ message complete their routing tables with the default route to the gateway. Although the problem of duplicated broadcast messages is solved with the same mechanism as in AODV ($RREQ$ ID), the periodical flooding of the whole network remains a big disadvantage of this approach. The cost of the flooding can not be ignored because of the limited resources in a mobile ad-hoc network.

Reactive Gateway Discovery

The mobile node that needs a route to a gateway broadcasts a $RREQ_I$. Only gateways react on this message by unicasting a $RREP_I$ back to the mobile node. Intermediate nodes only forward the $RREQ_I$. They prevent the duplicated forwarding with the known $RREQ$ ID mechanism. The advantage of this approach is that it is purely reactive. Only if routes to the gateway are



(a) Source node broadcasts a $RREQ_I$ **(b)** Gateway unicasts a $RREP_I$

Figure 3.8: AODV+: Reactive Gateway Discovery

needed, a request is started. Disadvantages are that the load on the forwarding node near the gateway is increased and the response time is higher.

Hybrid Gateway Discovery

The two approaches are combined. The gateway broadcasts periodically $RREP_I$ messages which are forwarded within the advertisement zone. Proactive gateway discovery is used for this area. The nodes outside of it have to start a reactive gateway discovery by sending a $RREQ_I$ message.

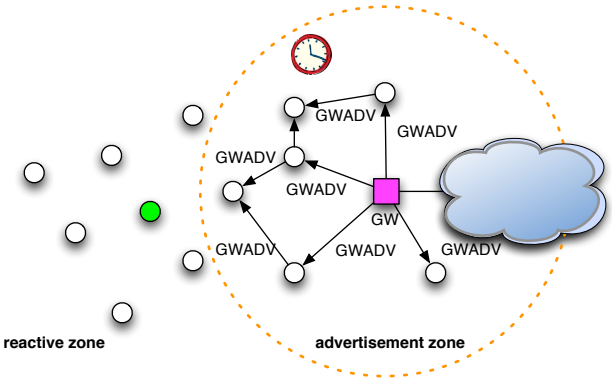


Figure 3.9: AODV+: Hybrid Gateway Discovery

Chapter 4

Cooperation in Mobile Ad-hoc Networks

In this chapter the need of cooperation in a MANET and various concepts of stimulating this cooperation are discussed. Furthermore, a closer look to the cooperation schemes CASHnet [16, 17, 18] and Nuglet [19] is taken.

4.1 Introduction

The field of application of MANET has become broader. Besides the use in emergency or military scenarios where all nodes belong to one authority, civilian and commercial scenarios are becoming more and more important today. Each node in the new scenarios forms its own authority. The nodes do not simply cooperate with each other by forwarding the packets. They act for their own profit. The individuality of the nodes causes selfishness. The selfishness of the nodes is expressed in the refusal of cooperation. For example, a node does not forward packets from other nodes in order to save its limited energy. It has no interest in spending its limited resources for the other nodes. It rather saves its resources to be able to send own packets. This behavior of the nodes leads to a malfunctioning MANET. It is no more possible to send packets over multiple hops. Therefore, the nodes have to be encouraged or forced to cooperate, in order to keep the MANET alive.

4.2 Concepts

There are different concepts to stimulate cooperation in a MANET or a multi-hop cellular network. The following two main concepts are distinguished, but certain cooperation schemes are using a mix of them.

4.2.1 Monitor and Punish Misbehavior

The activities of a node are monitored by a central authority or by the neighboring nodes. If a selfish behavior is detected, e.g. the node does not forward packets for the benefit of others, the node is punished. The node may be excluded from the network. It cannot send self-generated

packets to the network. Other actions are taken such as routing the packets around the non-cooperative node.

In the CONFIDANT [20] scheme cooperation is forced by detecting and isolating misbehaving nodes. Trust relationships and routing decisions are based on experienced, observed, or reported routing and forwarding behavior of the other nodes. A reputation system is included in each node. It maintains a local rating list. The monitoring is done by *neighborhood watch*, i.e. a node is observed by its one-hop neighbors. If a malicious or selfish node M is detected, the detecting node sends an $ALARM_M$ message to all its *friends*, i.e. trusted nodes, and adjusts the rating of M. A reception of an $ALARM_M$ message invokes also the reputation system of the node and causes an adaptation of the reputation level of the malicious node M. The paths for the packets are chosen regarding the nodes' ratings. Further, no route requests from malicious nodes are processed. This leads to an isolation of the malicious nodes from the network until their ratings are better. The problem is that a collective false accusation can exclude a node from the network without a fault of itself. The security is based on trust relationships and the reputation system.

In [21] the monitoring concept is combined with credits. The packet originator attaches a payment token to the packet. There is one payment per packet and not as in other schemes one payment per payee. The payment token can be thought of as a *lottery ticket*. Each intermediate node on the path verifies if the payment token is a winning ticket for it. The winning tickets are reported to the next base-station. This report includes also the identities of the neighboring nodes on the packet's path. The neighbor rewards encourage the node to forward the packet even if it does not receive a winning ticket. After the base-station has verified the validity of the payment token, it sends the packet to its destination over the backbone network. The accounting center compares the winning tickets reports and the payment tokens received from the base-station. It credits the claimant and its reported neighbors. Furthermore, it charges a usage fee to the packet originator's account. Moreover, the central authority audits the behavior of the nodes by analyzing the collected information received from the base-station.

4.2.2 Rewarding Cooperation

This concept gives the cooperative node a reward for its forwarding of the packets for the benefit of other nodes. The concept is closely connected to a credit (virtual currency) or micro payment system. The originator and/or the receiver of a packet have to pay for the packet transmission.

Rewards have been introduced in the Sprite [22] scheme as incentive for cooperation in a MANET. The nodes are connected to the *Credit Clearance Service* (CCS) via a wireless overlay network, e.g. GPSR. The security concept is based on a *Public Key Infrastructure* (PKI). A certificate identifies the individual nodes and the messages are digitally signed. Furthermore, the sender knows the full path to the destination from a secure ad-hoc routing protocol based on DSR. The full path is included in the message. The receiving nodes verify the signature and if they are on the path. If any condition is not satisfied, the packet is dropped. Otherwise, the node processes the packet and saves the receipt. The node reports its receiving and forwarding activities by sending these receipts to the CCS. According to the receipts the CCS determines the charge and credit to each node that participated in the transmission of a packet. The originator

of the packet has to pay for the transmission. The intermediate nodes are rewarded. The authors of [22] validated the system using a game-theoretic perspective.

The authors of [23] and [24] propose charging schemes for a multi-hop cellular network environment. Both schemes are based on centralized accounting and security mechanisms. They require complete routing information from the sender to the receiver, i.e. source routing. But source routing does not scale well under high node mobility. Furthermore, the schemes do not support cost sharing between sender and receiver.

[23] requires an existing *Authentication, Authorization and Accounting* (AAA) infrastructure. The node authenticates itself in the provider's network by using the AAA infrastructure. The authenticated node uses an on-demand source routing protocol to get the full path to the desired destination. It sends the packet equipped with its certificate, the digitally signed full routing path and the keyed hash value on itself and the destination. The intermediate node checks the signature and the certificate. If it is correct, the node computes the new value for the hash chain and forwards the packet. The last intermediate node acquires a digitally signed confirmation of the received amount of data from the destination. After that, it notifies the access point (AP) about this confirmation and the intermediate nodes involved in the transmission. The AP verifies the participation of the each node and the data amount. It charges a volume-based fee to the sender's account and credits the intermediate nodes for their forwarding behavior.

In order to participate in the network, a node in [24] has to register itself at the provider. It receives a long-term symmetric key. The sender node S has to establish an end-to-end session to the destination D. This is done by generating an *initiator session* between S and its base-station BS_S and a *correspondent session* between D and its base-station BS_D. Therefore, all communication passes through a base-station. The remuneration is done by charging the initiator S of the communication and rewarding the forwarding nodes. S pays for the traffic in both directions. The trusted provider of the base-stations maintains a billing account for every node in the multi-hop cellular network. The initiator is charged when the packet has passed through the base-station. The up-stream forwarding nodes are rewarded when the base-station has received an acknowledgment message that confirms the reception of the packet at the destination D. In order to motivate D to acknowledge the packet reception, the destination is charged a small fee ϵ when the packet is sent to the destination's network. ϵ is reimbursed when the acknowledgment is received at the base-station.

In the Nuglet [19] scheme cooperation in a MANET is enforced. A node can only send a self-generated packet, if it has earned enough credits by forwarding packets of other nodes. In contrast, the CASHnet [16, 17, 18] scheme lets the decision of participation in the multi-hop cellular network at the node's side, i.e. a node can even send self-generated packets, if he is not forwarding packets for others. But the node can earn credits by participation in the network, it can lower its costs for sending packets. The CASHnet and the Nuglet cooperation scheme are discussed in more details in the section 4.3.

Comparison

The tables 4.1 and 4.2 present a quick overview of the shown cooperation schemes. In table 4.1 the schemes are categorized according their architecture, security concept and their stimulation type. A decentral cooperation architecture is more suitable for a MANET because of its decentral

nature. Furthermore, a decentral cooperation architecture can also be preferable for a multi-hop cellular network. But it makes the security concept more complex. The security concept is either based on trust and reputation, symmetric session keys, PKI or AAA architecture. This choice has influences of the power used at the mobile node, e.g. the use of public / private key cryptography consumes a lot of processing power and therefore energy at the node.

Table 4.2 classifies the cooperation schemes according further criteria. The monitoring schemes observe the cooperation behavior of the nodes in contrast to the shown rewarding schemes. The schemes are initially targeted for a specific network type (MANET or a multi-hop cellular network). They have certain requirements concerning the routing protocol. The CASHnet and the Nuglet scheme requires only hop count information from the routing protocol in contrast to the other schemes that require source routing. They provide more flexibility in the choice of the routing protocol.

scheme	architecture	security concept	stimulation type
Nuglet [19]	decentral	PKI security associations message signing with symmetric session key	enforcement of cooperation (rewarding)
CASHnet [18]	decentral with service points	PKI message signing using public key	rewarding
CONFIDANT [20]	decentral	decentral trust relationships + reputation system	monitor / punishment of misbehavior
Sprite [22]	central authority overlay network	PKI message signing	rewarding (node reports forwarding activities to central authority)
[21]	central	symmetric session keys	monitoring (rewards / punishments)
[23]	central authentication / accounting	AAA architecture symmetric session keys hash chain	rewarding
[24]	central accounting	symmetric session keys all traffic is routed over base-station	rewarding

Table 4.1: Comparison of cooperation schemes I

4.3 Examples

4.3.1 CASHnet

The CASHnet [16, 17, 18] is targeted at multi-hop cellular network as shown in Fig. 4.1. The infrastructure, such as the gateways and the service stations, is offered by a provider. The CASHnet scheme provides decentralized accounting and security mechanisms in this multi-hop cellular

scheme	monitoring	routing information	target network
Nuglet [19]	no	hop count	MANET
CASHnet [18]	no	hop count	multi-hop cellular network
CONFIDANT [20]	yes neighbors monitor the node	full path reactive source routing DSR	MANET
Sprite [22]	no	full path source routing DSR	MANET
[21]	yes. central authority	source routing DSR	multi-hop cellular network
[23]	no	source routing	multi-hop cellular network
[24]	no	source routing	multi-hop cellular network

Table 4.2: Comparison of cooperation schemes II

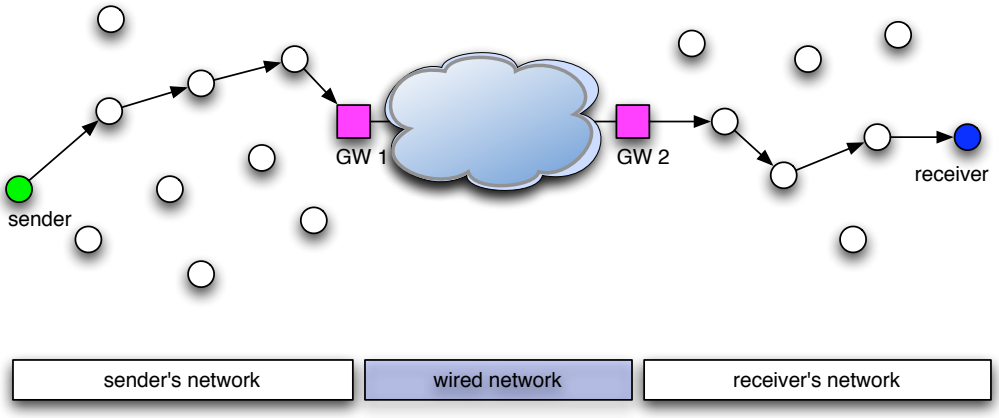


Figure 4.1: CASHnet: Example scenario

network. It supports sender- and receiver-based payments, and does not require complete route information from the sender to the receiver. It coexists with pure ad-hoc traffic. The nodes are neither charged nor remunerated for this ad-hoc only traffic. Because pure ad-hoc communication does not cause any costs in terms of bandwidth for the provider, it should be free of charge. By providing monetary rewards to cooperative nodes, cooperation becomes a gainful alternative to selfishness for the nodes. It is a rewarding scheme.

The security mechanisms are implemented in a tamper resistant device, e.g. a smart card. They rely on a *Public Key Infrastructure* (PKI) system. This security service could be charged as a subscription fee in order to receive the smart card. Besides the gateways, the provider also offers service stations where the users can trade earned Helper Credits and buy further Traffic Credits. The service stations could be integrated in the gateways as well as additionally installed within the ad-hoc network.
CASHnet requires the following components:

- A tamper resistant device, e.g. a smart card, that provides a protected environment to execute safely the CASHnet functions. It contains the counters for Traffic Credits and Helper Credits, the node's public / private key pair and the node's certificate, the provider's public key and two internal lists (Reward and Authentication List). Each node needs such a tamper resistant device. It is sold by the provider as todays SIM cards for GSM mobile telephony.

- A routing algorithm that provides hop count information (e.g. AODV or DSR).

- Gateways that route the messages between the MANET and the wired backbone of the provider.

- service stations of the provider. There, the nodes can buy Traffic Credits, trade Helper Credits, or renew their certificate. The service stations can be imagined as todays ATM.

The security architecture of CASHnet is based on public key cryptography. The provider issues certificates to the nodes. The nodes authenticate themselves with these certificates. The certificates have a short lifetime to avoid the creation of bogus nodes. This fact forces the nodes to regularly visit one of the provider's service station to renew it. All data messages are digitally signed to guarantee non-repudiation, which provides data integrity and data origin authentication.

The charging and rewarding mechanism of CASHnet in the sender network is shown in numbered actions in Fig. 4.2. If node S wants to send a self-generated packet outside the ad-hoc network, it has to pay with Traffic Credits. The costs are proportional to the hop count to the gateway. Every intermediate node that forwards the packet gets Helper Credits. As already mentioned, Traffic Credits can be bought for real money or traded for Helper Credits at a service station. After paying for the packet transmission (action 1), the packet can be sent to the next node (action 2). The first intermediate node forwards the packet without sending an acknowledgment message (ACK) because the originator of the packet should not be rewarded (action 3). The following nodes confirm the reception of the packet by sending an acknowledgment message (ACK) to the precursor node (actions 4, 7, 10) and forward the packet to the next hop

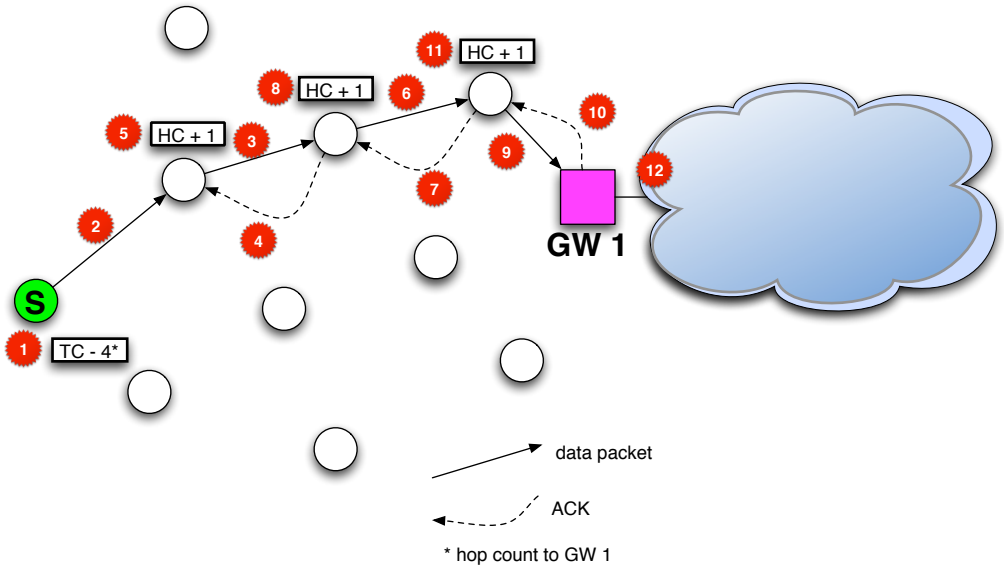


Figure 4.2: CASHnet: Node S sends a packet outside the ad-hoc network.

(actions 6, 9). The use of the acknowledgment mechanism ensures the proper delivery of the packets. The precursor node is rewarded by this ACK message with Helper Credits (actions 5, 8, 11). The gateways provide the interconnection of the two ad-hoc networks. A subscription fee can compensate the costs of this service. The packet is sent through the interconnection network to the other gateway (action 12).

In the receiver's network (numbered actions in Fig. 4.3) the intermediate nodes forward the packet towards the destination D (actions 13, 16, 19, 22). The nodes in the receiver's network acknowledge the reception of the forwarded packet to their precursors (actions 14, 17, 20, 23). The precursors earn Helper Credits when they receive the ACK message (actions 15, 18, 21, 24). But there, the receiver of the packet has to pay the costs for the packet proportional to the hop count in order to receive the packet (action 25).

The operation process of CASHnet is divided in six different phases. They are shown for a multi-hop cellular network that belongs to provider P. In the *Setup Phase*, the node N has to perform the following tasks to be able to participate in the network:

1. The node needs to get a personal tamper resistant device (e.g. smart card) from the provider P. The device contains the node N's unique ID_N, the public / private key pair K_N/KP_N, a certificate $Cert_P(ID_N, K_N)$ issued by the provider P, and the public key K_P of the provider P. This task is only performed once.

2. If necessary, the node N's certificate $Cert_P(ID_N, K_N)$ has to be updated.

3. If necessary, the Traffic Credit account has to be loaded. This can by done at the provider P's service station either by paying with real money and/or by trading Helper Credits.

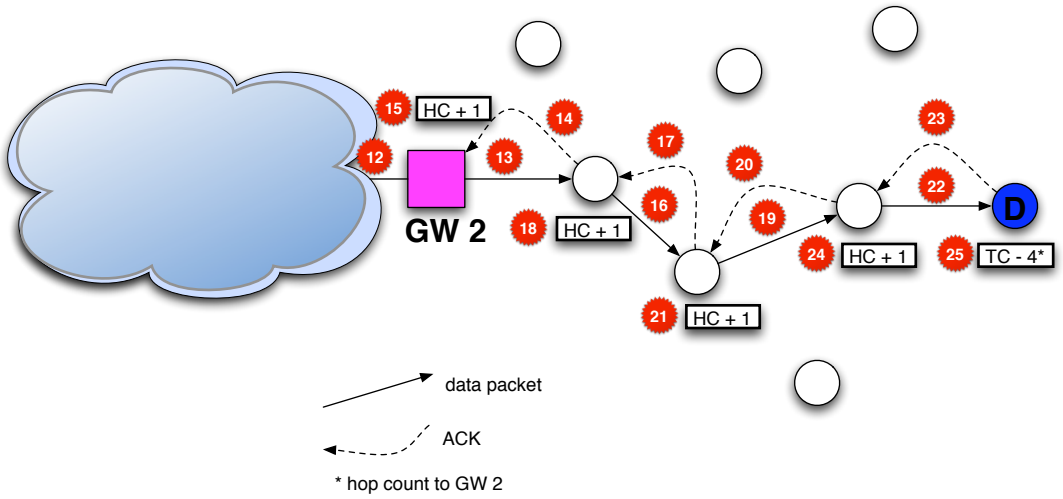


Figure 4.3: CASHnet: Node N receives a packet

In the *Initial Authentication Phase*, the node S that wants to send a packet to the destination D has to build up an authenticated route to D. The node sends out an authentication request (AUTH REQ) to the destination. This AUTH REQ message includes S's ID_S, its public key K_S, and the certificate $Cert_P(ID_S, K_S)$. The intermediate nodes verify the certificate. If it is valid, the nodes include S's identity ID_S and public key K_S in their AUTH list. The destination D sends back an authentication reply message (AUTH REP) to the sender S, after the reception of a valid AUTH REQ. When the node S receives the AUTH REP, a route with cooperative nodes exists and it can begin to transmit self-generated data packets. The process is illustrated in Fig. 4.4. Every intermediate node N that participates in a transmission has to authenticate itself to the previous and next node on this active route. Therefore, a periodic authentication to the one-hop neighbors of the node N as shown in Fig. 4.5 is introduced to reduce the delay caused by unauthenticated nodes on the forwarding path. The successfully authenticated neighboring nodes are also stored in the AUTH list.

The *Packet Generation Phase* covers the steps that a node S processes to send a data packet to the destination node D. It is shown in Fig. 4.6(a). The data packet $Packet_S$ has the format $ID_S|Payload|Timestamp_S|Sig_S(Payload, Timestamp_S)$. The node S has to pay a transmission fee to send the packet P_S (sender-based payment).

The reception of a data packet P_{N-1} is treated in the *Packet Reception Phase* as illustrated in Fig. 4.6(b). The node N receiving a data packet $Packet_{N-1}$ from the node $N-1$ acts as described in the flowchart. The acknowledgment message ACK_N has the format $ID_N|Timestamp_N|Sig_N(Sig_{N-1}, Timestamp_N)$. If the data packet P_N reaches its destination, the destination R has to pay the reception fee (receiver-based payment) to receive the packet.

Fig. 4.7(a) shows the *Packet Forwarding Phase*. The node N forms a new packet P_N which includes the original packet P_S, the new time stamp, and a signature of N. The format of packet P_N is $ID_N|P_S|Timestamp_N|Sig_N(P_S, Timestamp_N)$. The new packet P_N is sent to the

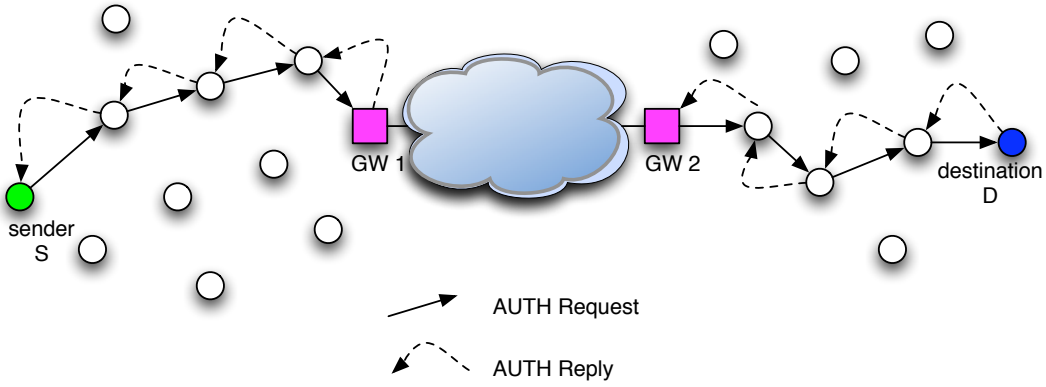


Figure 4.4: CASHnet: Building up an authenticated path between the nodes S and D

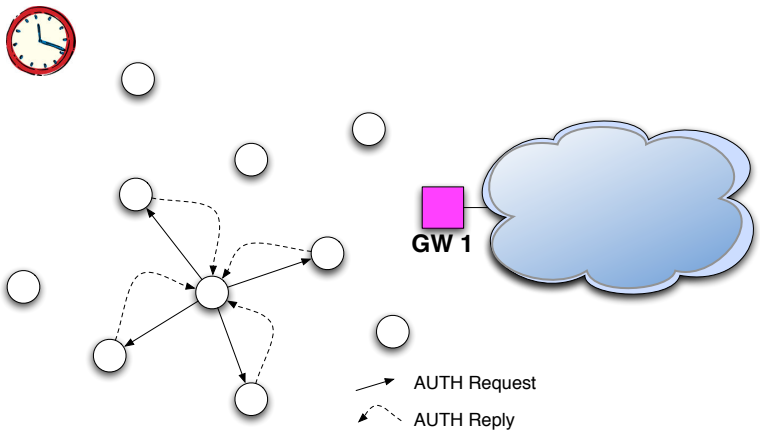


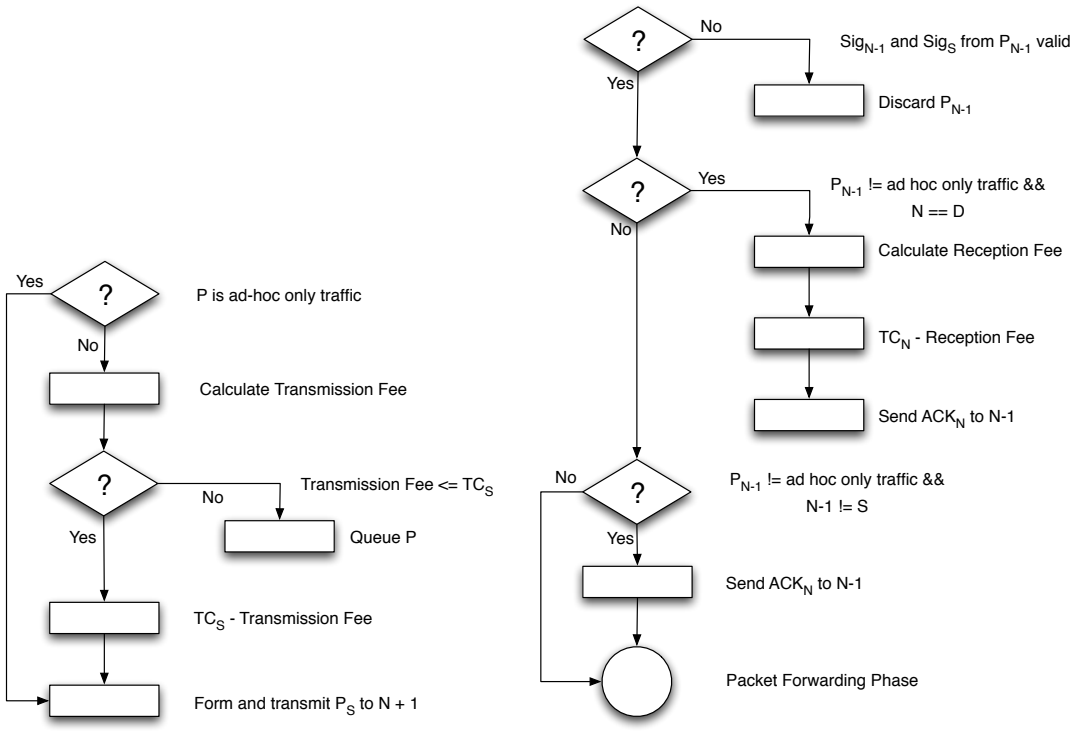
Figure 4.5: CASHnet: Periodic authentication to one-hop neighbors

next hop toward the destination D.

Finally, the cooperative nodes receive their benefit in the *Rewarding Phase*. The reward is added to their Helper Credit account. These Helper Credits can be traded against Traffic Credits at a service station of the provider P. To guarantee authorized reward only, the acknowledgment messages ACK_{N+1} are signed, and the pair $< Sig_N(P_N), ID_{N+1} >$ must be included in the reward list of the node. The reward list is protected by the tamper resistant device.

4.3.2 Nuglet

The Nuglet scheme [19] offers decentralized accounting and security mechanisms in MANETs. It provides sender-based payment. In contrast to the CASHnet scheme, where cooperative nodes can save money, it forces the nodes to cooperate by making the ability to send own packets dependent on the cooperativeness of the node. A node has to forward packets for the benefit of others in order to get credits to send self-generated packets.



(a) Packet Generation Phase **(b)** Packet Reception Phase

Figure 4.6: CASHnet: Phases I

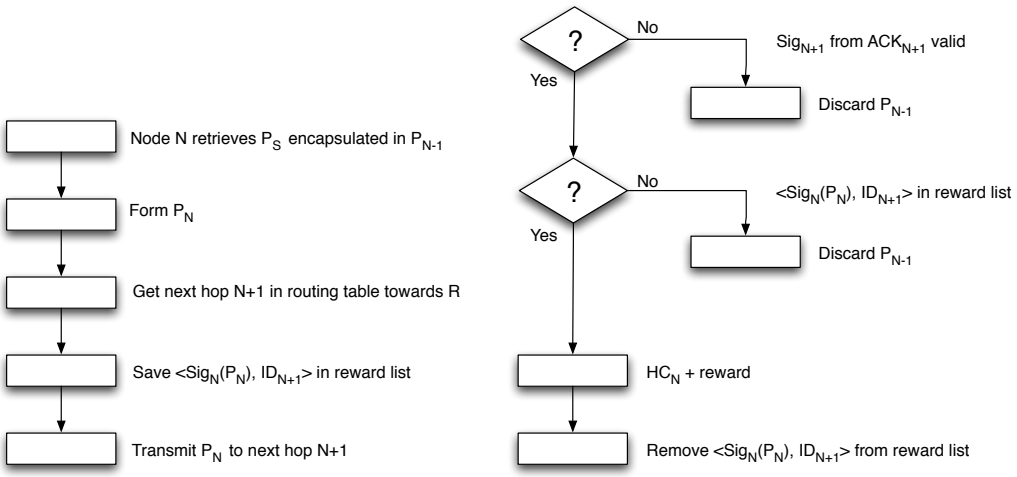
Nuglet requires the following components:

- A tamper resistant security module A, e.g. a smart card or a cryptographic coprocessor [25], that offers a protected environment for the routing and the Nuglet functions. It contains a credits counter, so called *Nuglets counter* (nc), the public / private key pair of the node, a certificate $Cert_A$, the manufacturer's certificate $Cert_M$, a list of all certificates of manufacturer's of such security modules (assumption that only a few of them exists), the unique identifier of the security module ID_A, a protected database of the security associations, a list with the *pending counters* $pc_{X@A}$ for each neighbor node.

- A routing algorithm, that allows estimation about the number of intermediate nodes toward a destination. It runs in the protected environment of the security module.

Nuglet's operation is based on the following two rules (see example in Fig. 4.8) :

- When a node wants to send self-generated packets, it has to pay for them with Nuglets. The costs of a packet are the estimated number of intermediate nodes (ni) on the route to the destination. The costs of a transmission (ni) are subtracted form the *Nuglets counter*



(a) Packet Forwarding Phase **(b)** Rewarding Phase

Figure 4.7: CASHnet: Phases II

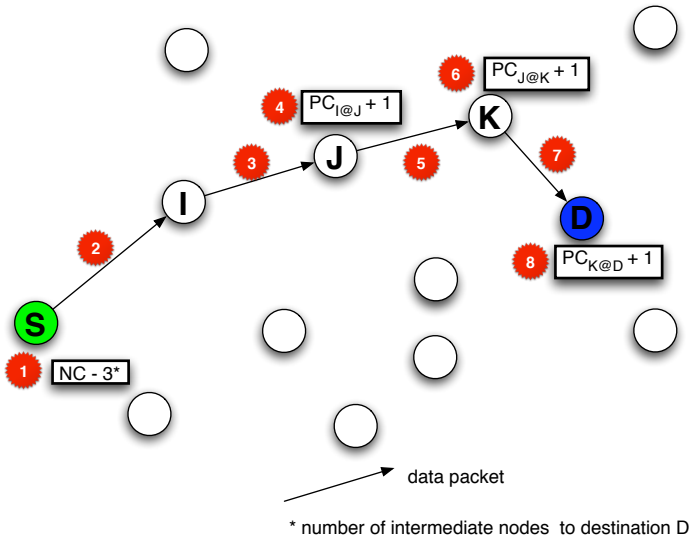


Figure 4.8: Nuglet: Node S sends a packet to the destination node D.

(nc). Because the nc can not be negative, a node can only send an own packet, if the nc is greater than or equal to ni. Otherwise, the packet cannot be sent, and the nc stays the same.

- By forwarding other nodes' packets, a node can gain Nuglets, i.e. increase its nc. These

Nuglets are not credited directly to the nc, they are store in the next node's *pending counter* ($pc_{N@N+1}$) and periodically transmitted by a synchronization protocol.

Of course, the stimulation mechanism of Nuglet has to be protected against different attacks. Nobody should be able to manipulate the *Nuglets counter*, especially increasing it. Further, it must be ensured that the *Nuglets counter* is only increased if the forwarding packet is really forwarded. The manipulation of the *Nuglets counter* is prevented by using the above mentioned security module. All critical functions are included in it. It is ensured that the node cannot gain any advantages by the manipulation of unprotected functions. The rewarding is ensured by the usage of the *pending counters*.

Nuglet uses a public-key infrastructure (PKI). Each security module has a public/private key pair. The public key is certified by the manufacturer of the module. The number of manufacturer is limited and all of them cross-certify their public keys.

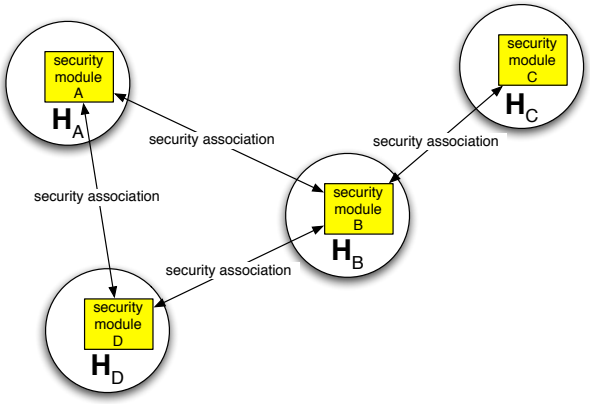


Figure 4.9: Nuglet: Neighboring nodes establish security associations.

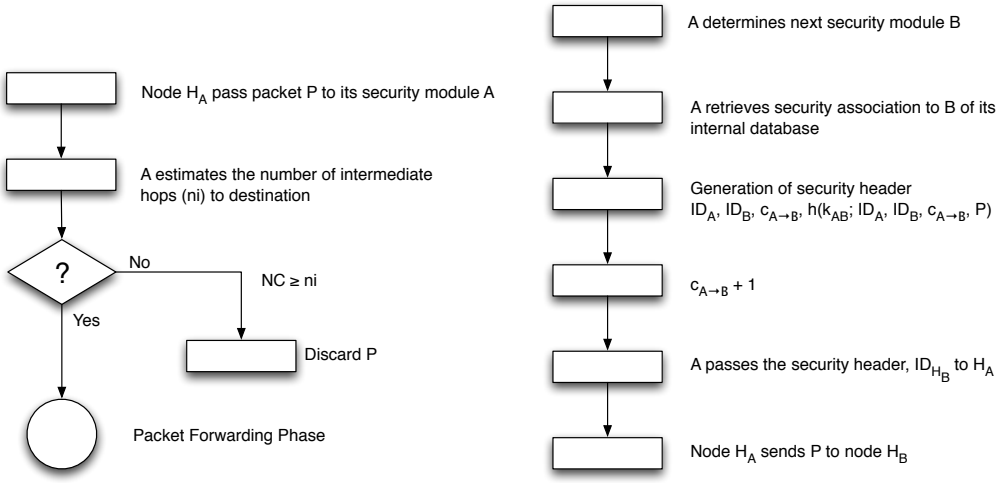
The security modules of neighboring nodes establish symmetric key sessions, so called security associations, between each other (see Fig. 4.9). If these associations cannot be built, the routing protocol does not consider them neighbors. A security association between two security modules A and B contains at A's side:

- the unique identifier ID_B of the other security module B
- the unique identifier ID_{H_B} of the node H_B that hosts B
- the symmetric session key k_{AB}
- the *sending* and *receiving sequence numbers* $c_{A \rightarrow B}$, $c_{A \leftarrow B}$
- the *pending Nuglets counter* $pc_{B@A}$

The integrity of the packet and its authenticity between the security modules A and B is protected by a message authentication code generated with the session key k_{AB}. It is also possible

to provide a link-by-link encryption by using k_{AB}. Packet replay attacks are detected by the usage of the increasing sequence numbers $c_{B\rightarrow A}$, $c_{A\leftarrow B}$, $c_{B\rightarrow A}$, and $c_{B\leftarrow A}$. The establishment of the security association is done by using some public-key cryptographic protocol. The hosting nodes of the security modules run this protocol.

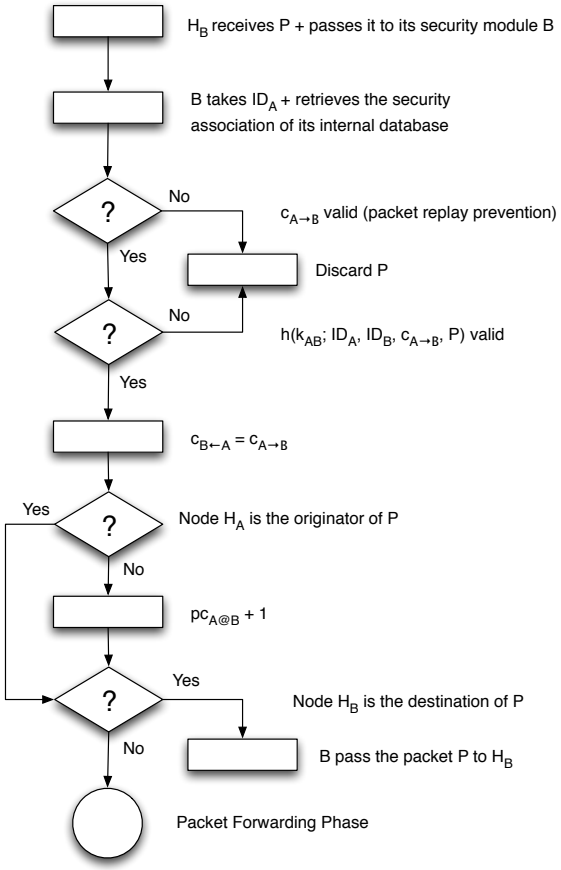
The operation of Nuglet is divided into different phases in order to easily compare it with the CASHnet scheme. The *Setup Phase* consists of getting a security module with the corresponding keys and certificates from a manufacturer. In the Authentication Phase, the security associations are built up. The *Packet Generation Phase*, the *Packet Reception Phase* and the *Packet Forwarding Phase* are illustrated in Fig. 4.10(a), 4.10(b) and 4.11(a).



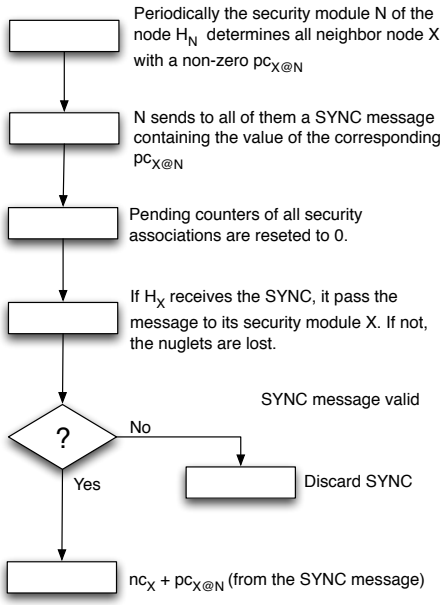
(a) Packet Generation Phase **(b)** Packet Forwarding Phase

Figure 4.10: Nuglet: Phases I

The cooperative nodes in the network receive their credits in the periodically executed *Nuglet Synchronisation Phase* (see Fig. 4.11(b), 4.12). The Nuglets in *pending counters* stored in the security association of a node are transmitted to their owners. This concept is introduced to ensure that the packets are really forwarded.



(a) Packet Reception Phase



(b) Packet Synchronisation Phase

Figure 4.11: Nuglet: Phases II

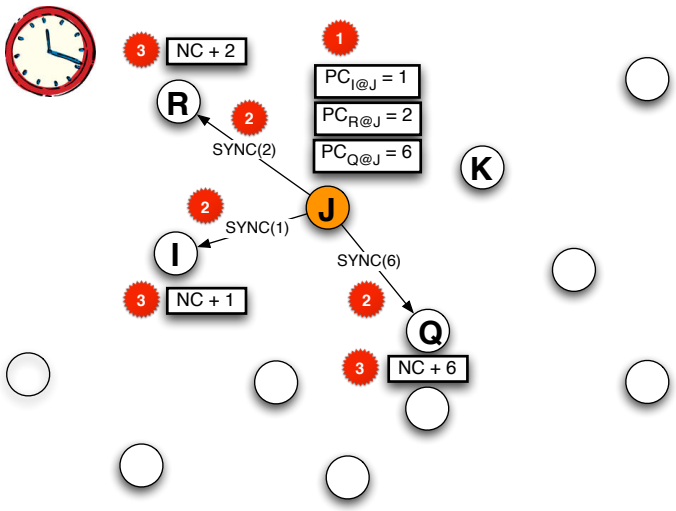


Figure 4.12: Nuglet: Node N submits the *pending counters* $pc_{X@N}$ in $SYNC$ messages to their owners.

Chapter 5

Implementation of Simulation Model

This chapter describes the network simulator 2 (ns2) [15], its tracing mechanism and especially the wireless model in ns2. In addition, a closer look to the used Random Way Point Mobility model is taken. Finally, the implementations of CASHnet and Nuglet in ns2 are discussed.

5.1 Background and Requirements

In order to verify the CASHnet cooperation scheme an implementation in a network simulator is chosen. The alternative of an implementation in a real system (e.g. Linux) and testing it as experimentation would use too much resources and finally be too expensive. Furthermore, the implementation in a simulator offers more flexibility and variations, i.e. scenarios with much more nodes can be tested and adapted for the initial parameter tuning. An implementation in real systems can be considered, if the verification with the help of the simulation is successful. A network simulator for the verification of the cooperation schemes should fulfill the following requirements:

- Simulation scenarios with 50 and more nodes.
- Gateways which are able to route packets between the wired and the wireless networks.
- Ad-hoc routing protocol that can supply the hop count for the cooperation scheme (CASHnet or Nuglet).
- Ad-hoc routing protocol that is able to detect the gateways.
- Physical Layer model with Radio Propagation.
- MAC Layer and Link Layer models.
- Mobility of the nodes.
- Enhanced tracing functionality.

There exists quite a number of network simulators today. Not all of them have a good reputation within the research community, and of those which have, most are expensive. Therefore, ns2

[15] is chosen, because it is open source software, freely available and it is widely used in the research community. Besides, ns2 meets perfectly the requirements. The protocol stack for MANET and multi-hop cellular networks is implemented. With AODV+ [13], there exists also a mobile ad-hoc routing protocol that is aware of gateways.

5.2 Implementation with Network Simulator 2

Ns2 [15] is a discrete event driven simulator. The source code and the documentation [26] are currently maintained by the Virtual Internet Testbed (VINT) at the Information Sciences Institute (ISI) of the University of Southern California (USC). The goal of ns2 is to support networking research and education. It provides an environment for protocol design, traffic studies and protocol comparison. Its license model enables the sharing of code, protocols, models, and ensures that the work is given back to the community. It allows easy comparison of similar protocols. This collaborative environment and the big number of users should also increase the confidence in the results because more people look at the models in more situations than by using a closed source simulator.

5.2.1 Structure of ns2

In ns2 real world objects are modeled by objects in the simulation and programmed to react as much as possible as their correspondents in the real world would react. In the concept of event driven simulation, physical activities are translated to events. The events are stored in a queue. They are processed in the order of their scheduled occurrences. The time in the simulation progresses as the events are processed. Each event happens in an instant of simulated time, but takes an arbitrary amount of real time. Ns2 is built using object oriented methods in C++ and OTcl (see Fig. 5.1). The developers of ns2 tried to combine fast iteration time with good run-time

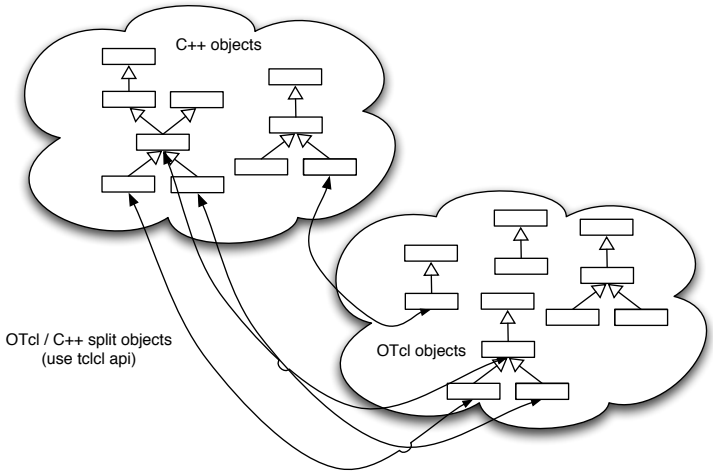


Figure 5.1: Duality of C++ and OTcl in ns2

performance. This results in a mixed coding framework in C++ and OTcl. C++ serves as system programming language in which all time consuming components, e.g. packet processing and routing algorithms, are implemented. OTcl is used as the configuration language for the simulation scenarios. It allows the quick setup of different simulation scenarios and an interactive simulation mode. OTcl and C++ share linked class hierarchies and the additional library TclCL offers sharing of functions and variables.

Objects in C++ are compiled and then made available to the OTcl interpreter through an OTcl linkage (TclCL) which maps methods and member variables of the C++ object to methods and variables of the linked OTcl object. This system architecture facilitates the usage of ns2 and its existing components, but it makes the development of new components complicated and time-consuming.

Internal Packet Representation

The internal packet representation of ns2 is quite different from a packet in the real world. The packet in the simulator contains all headers that the simulator supports, e.g. UDP, TCP, MAC, IP etc., and not only the headers of the real world packet. Furthermore, a packet in the simulator has a common header which contains important simulation information, e.g. the simulated packet size (i.e. size of the real world packet), the packet type, the flow direction, a unique packet ID and a time-stamp. Besides, the packet headers of ns2 do not necessary correspond to the protocol headers defined in RFCs, e.g. header checksums are normally left out.

Simulation Process

Fig. 5.2 shows the simplified process for a simulation. The user has to set the different components, e.g. event scheduler objects, network components and setup module libraries, up in the simulation environment. This is done by a simulation script in OTcl. The script is processed by ns2 and delivers trace files that the user analyzes with the Network Animator (NAM) or custom scripts.

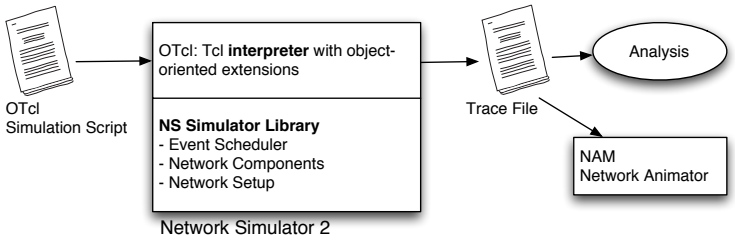


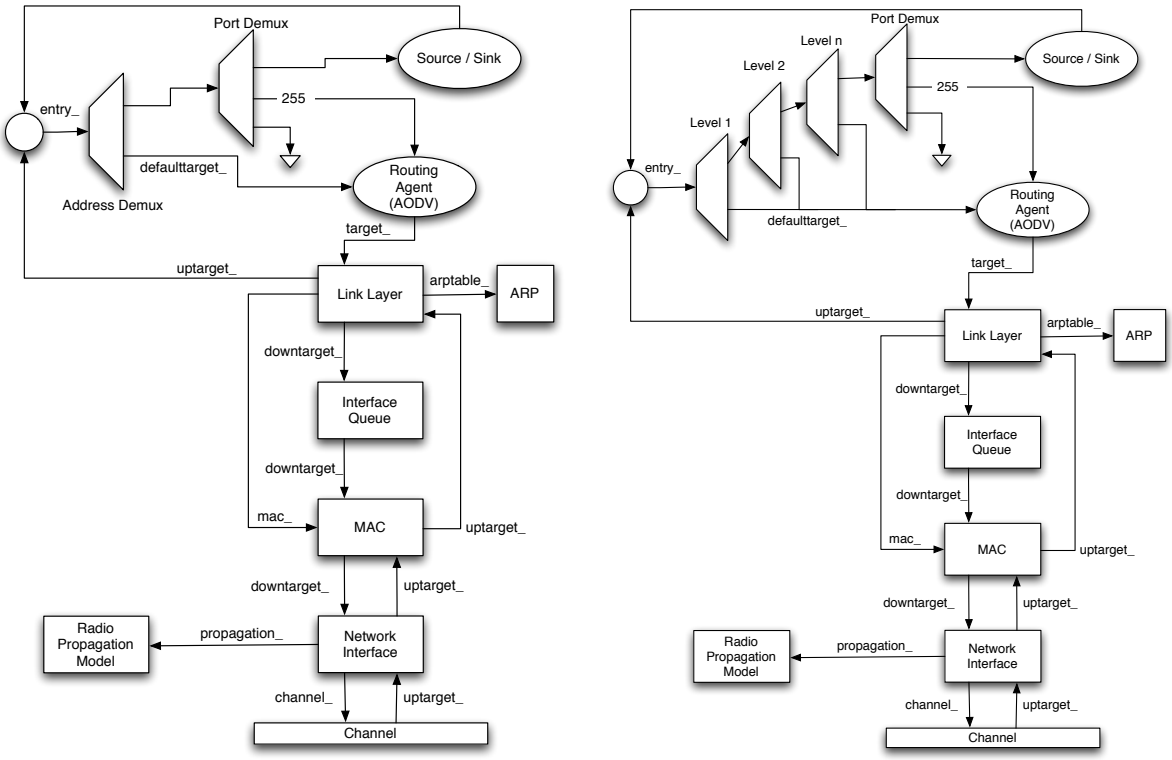
Figure 5.2: User's view of ns2

5.2.2 Wireless Model in ns2

The wireless model in ns2 is contributed from CMU's Monarch project (Wireless extension to ns2). Various modules were added to ns2 to simulate node mobility and wireless networking.

- Mobile Node
- Basestation Node
- Ad-hoc Routing Agents (DSR, DSDV, TORA, AODV, AODV+)
- MAC 802.11
- Radio Propagation Model
- Channel

The Fig. 5.3(a) and 5.3(b) show the structure of a mobile node in ns2. The variables ending with an underscore, e.g. *entry_*, mark points accessible within the OTcl interpreter.



(a) with normal addressing **(b)** with hierarchical addressing

Figure 5.3: Representation of *MobileNode*

The mobile node consists of the following components:

Address Classifier examines the packets destination field and forwards the packet to the right component in the mobile node. If the node supports hierarchical addressing (e.g. needed

in multi-hop cellular networks), the address classifying component consists of multiple address classifiers (see Fig. 5.3(b)).

Port Classifier classifies the packets destination port and forwards the packet to the correct receiving Agent on a node, e.g. AODV control packets are forwarded to port 255 where the AODV routing agent is listening.

Agent is responsible for packet generation and reception, similar to an application layer program. There exist various Agents such as CBR (Constant Bit Rate), TCP, FTP, etc.

Link Layer runs the link layer protocols. It fragments and reassembles the packets. It runs the Address Resolution Protocol (ARP) to resolve IP addresses to MAC addresses.

Interface Queue gives priority to routing protocol packets.

MAC (Media Access Control) is implemented as IEEE 802.11 protocol.

Network Interface is the hardware interface used by the mobile node to access the wireless channel. Here the signal integrity, collisions, and transmission errors are simulated. Each transmitted packet is marked with transmission power, wavelength etc.

Radio Propagation Model uses Friss-space attenuation ($1/r^2$) at near distance and Two Ray Ground Model ($1/r^4$) at far distance. It implements an omni-directional antenna which has an unity gain for all directions. It checks if an simulated packet can be received with the transmission power and wavelength set in the packet and the given distance to the sender of the packet.

Channel takes packets from the network interface and copies them to all other network interfaces.

The mobile nodes are configured using the following *node-config* interface.

```
$ns_ node-config        -addressType flat/hierarchical
                        -adhocRouting   DSDV/DSR/TORA/AODV
                        -llType
                        -macType
                        -propType
                        -ifqType
                        -ifqLen
                        -phyType
                        -antType
                        -channel
                        -channelType
                        -topologyInstance
                        -wiredRouting   ON/OFF
                        -mobileIP       ON/OFF
                        -energyModel    "EnergyModel"
                        -initialEnergy  (in Joules)
                        -rxPower        (in W)
                        -txPower        (in W)
                        -idlePower      (in W)
                        -agentTrace  ON/OFF
                        -routerTrace ON/OFF
                        -macTrace ON/OFF
                        -movementTrace ON/OFF
```

5.2.3 Tracing

Ns2 offers tracing of all packets in the simulation. Furthermore, ns2 enables the tracing of variables in C++ or OTcl and supports the monitoring of queues and flows (see [26] for detailed information). In this thesis only the packet tracing ability is used. There exist three different trace file formats (*old*, *new wiressless* and *NAM*) for packet tracing. [27] gives a good overview of them. In this thesis the trace files are generated in the new wireless trace format. The new wireless trace format is defined in the source files $trace/cmu-trace.\{h \mid cc\}$. The usage of the new trace file format requires the following lines in the simulation script:

```
$ns_ use-newtrace
set tracefile [open <path of tracefile >w]
$ns_ trace-all $tracefile
```

A line of the trace file starts with an action flag (see Tab. 5.1(a)) which specifies the action that the node has performed on the packet. Then multiple flag / value pairs follow. A value flag consists of a $-$, followed by one character indicating the type (see Tab. 5.1(b)) and one or two additional characters (Tab. 5.2, 5.3).

s	send
r	receive
d	drop
f	forward

(a) Action flags specify the action that was processed to the packet

N	Node Property
I	IP Level Packet Information
H	Next Hop Information
M	MAC Level Packet Information
P	Application Level Packet Information

(b) Flag types for the new wireless trace format

Table 5.1: New wireless trace format part I

Example

Four lines are picked of a trace file as an illustration (see listing 5.1 on p. 38). The first line shows a reception of a constant bit rate (CBR) packet at node 2. The second line illustrates the sending of a CBR packet at the time 71.01. In line 3 an AODV $RREQ$ packet is dropped at node 8 because of an expired TTL field (*-Iv 0*, drop reason *-Nw TTL*). Finally, the line 4 describes a forwarded AODV $RREP$ packet at the node 18.

Listing 5.1: Four Example lines of a trace file with the new wireless trace file format

```
r -t 46.189458524 -Hs 2 -Hd 8388608 -Ni 2 -Nx 1500.00 -Ny 400.00 -Nz 0.00 -Ne -1.000000 -Nl AGT -Nw --- -Ma 13a -Md
      1 -Ms 16 -Mt 800 -Is 4194341.0 -Id 8388608.36 -It cbr -Il 532 -If 1 -Ii 446 -Iv 28 -Pn cbr -Pi 9 -Pf 3 -Po 0
s -t 71.010000000 -Hs 26 -Hd -2 -Ni 26 -Nx 174.03 -Ny 306.05 -Nz 0.00 -Ne -1.000000 -Nl AGT -Nw --- -Ma 0 -Md 0 -Ms
      0 -Mt 0 -Is 4194328.0 -Id 8388608.23 -It cbr -Il 512 -If 1 -Ii 1127 -Iv 32 -Pn cbr -Pi 14 -Pf 0 -Po 0
d -t 74.661563491 -Hs 8 -Hd -2 -Ni 8 -Nx 758.05 -Ny 571.11 -Nz 0.00 -Ne -1.000000 -Nl RTR -Nw TTL -Ma 0 -Md ffffffff
      -Ms 17 -Mt 800 -Is 4194310.255 -Id -1.255 -It AODV -Il 48 -If 0 -Ii 0 -Iv 0 -P aodv -Pt 0x2 -Ph 4 -Pb 13 -Pd
      4194343 -Pds 29 -Ps 4194341 -Pss 34 -Pc REQUEST
f -t 86.411356790 -Hs 18 -Hd 4194328 -Ni 18 -Nx 353.16 -Ny 422.37 -Nz 0.00 -Ne -1.000000 -Nl RTR -Nw --- -Ma 13a -Md
      11 -Ms 24 -Mt 800 -Is 4194320.255 -Id 4194328.255 -It AODV -Il 44 -If 0 -Ii 0 -Iv 29 -P aodv -Pt 0x4 -Ph 8
     -Pd 8388608 -Pds 70 -Pl 9.000000 -Pc REPLY
```

Flag	Type	Value
	s—r—d—f	action type
-t	double	Time
-Ni	int	Node ID
-Nx	double	X Coordinate
-Ny	double	Y Coordinate
-Ne	double	Node Energy Level
-Nl	string	Trace Name (AGT, RTR ...)
-Nw	string	Drop Reason
-Hs	int	Node ID
-Hd	int	Node ID For Next Hop
-Ma	hexadecimal	Duration
-Ms	hexadecimal	Source Ethernet Address
-Md	hexadecimal	Destination Ethernet Address
-Mt	hexadecimal	Ethernet Type
-P	string	Application Type (arp, dsr, cbr, tcp, ...)

Table 5.2: New wireless trace format part II

Event	Flag	Type	Value
ARP Trace	-Po	string	Request or Reply
	-Pms	int	Source MAC Address
	-Ps	int	Source Address
	-Pmd	int	Destination MAC Address
	-Pd	int	Destination Address
AODV Trace	-Pt	hexadecimal	Type
	-Ph	int	Hop Count
	-Pb	int	Broadcast ID
	-Pd	int	Destination
	-Pds	int	Destination Sequence Number
	-Ps	int	Source
	-Pss	int	Source Sequence Number
	-Pl	double	Lifetime
	-Pc	string	Operation (REQUEST, REPLY, ERROR, HELLO)
IP Trace	-Is	int.int	Source Address and Port
	-Id	int.int	Destination Address and Port
	-It	string	Packet Type
	-Il	int	Packet Size
	-If	int	Flow ID
	-Ii	int	Unique ID
	-Iv	int	TTL Value
TCP Trace	-Ps	int	Sequence Number
	-Pa	int	Acknowledgment Number
	-Pf	int	Number of Times Packet was forwarded
	-Po	int	Optimal Number of Forwards
CBR Trace	-Pi	int	Sequence Number
	-Pf	int	Number of Times Packet was forwarded
	-Po	int	Optimal Number of Forwards

Table 5.3: New wireless trace format part III

5.3 Random Waypoint Mobility Model

The Random Waypoint Mobility Model is used to model the movements of the mobile nodes in the simulations in this thesis. This mobility model functions as follows. A mobile node begins the simulation by waiting a specified pause-time. After this time it selects a random destination in the area and a random speed distributed uniformly between *0* m/s and V_{max}. After reaching its destination point, the mobile node waits again pause-time seconds before choosing a new way point and speed.

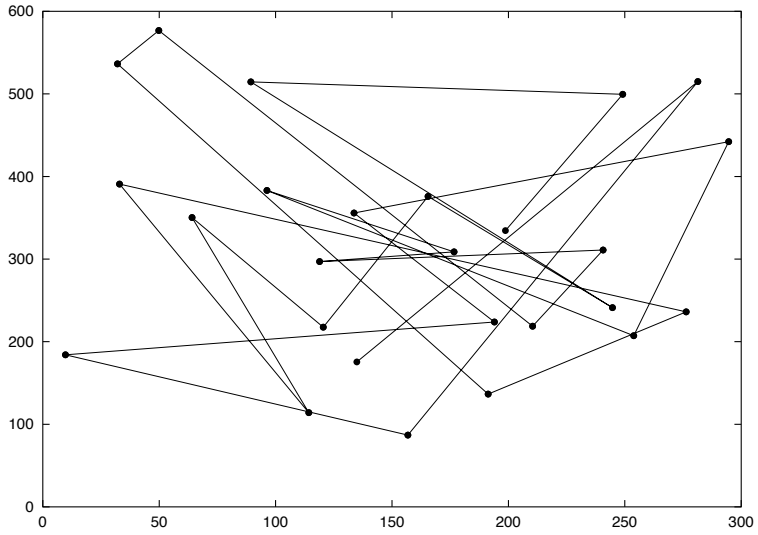


Figure 5.4: Traveling pattern of a mobile node using Random Waypoint Model (Fig. copied from [1])

The mobile nodes are initially distributed over the simulation area. This distribution is not representative to the final distribution caused by node movements. To ensure a random initial configuration for each simulation, it is necessary to register the simulation after some time has elapsed.

The Random Waypoint Mobility Model is very frequently used in simulation studies of MANET. As described in [28] the performance measurements in mobile ad-hoc networks are affected by the used mobility model. One of the most important parameters in mobile ad-hoc simulations is the nodal speed. The users of the simulator want to adjust the average speed to be stabilized around a certain value and not to change over time. They also want to be able to compare the performance of the mobile ad-hoc routing protocols under different nodal speeds. For the Random Waypoint Mobility Model a common expectation is that the average is about half of the maximum, because the speeds in a Random Waypoint Model are chosen uniformly between 0 m/s and V_{max}. The studies in [28] contradict this expectation and show that the average speed is decreasing over time and will approach 0.

This phenomenon can be intuitively explained as follows. In the Random Waypoint Mobility Model a node selects its destination and its speed. The node keeps moving until it reaches its destination at that speed. If it selects a far destination and a low speed around 0 m/s, it travels

for a long time with low speed. If it selects a speed near V_{max} the time traveling with this high speed will be short. After a certain time the node has traveled much more time at low speed than at high speed. The average speed will approach 0 m/s. The suggestion in [28] to prevent this problem is choosing 1 m/s instead of 0 m/s as V_{min}. With this approach the average speed stabilizes after a certain time at a value below $1/2 * V_{max}$.

5.4 CASHnet Implementation

A simplified CASHnet scheme without the security mechanism, but with the full charging and rewarding functionality, has been implemented in ns2. Because of the mixed coding in ns2, it is rather difficult to overview the internal packet flow in ns2. Methods and variables can be defined in C++ and made available to the OTcl interpreter. In addition, the source code is not well structured and the multiple inheritances in C++ complete the confusion. It is also very difficult and time-consuming to investigate in which files of the source code certain functionalities are implemented. But finally, the implementation becomes feasible with the help of the ns documentation [26] and the C++ class hierarchy on the ns2 web page.

Because there are similarities between the CASHnet and the Nuglet scheme a common super class named *CooperationNode* is generated. It extends the *MobileNode* class and provides the common interface for *CashnetNode* and *NugletNode* (see Fig. 5.5). Fig. 5.6 shows the structure of *CashnetNode*.

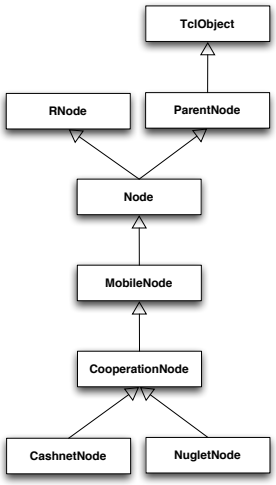


Figure 5.5: Class Hierarchy of *CooperationNode*, *CashnetNode* and *NugletNode*

The new node for the CASHnet scheme contains an agent to handle the ACK messages. The *CashnetACK_Agent* (Fig. 5.7(a)) sends and receives the new generated ACK messages.

The routing agent AODV from the AODV+ package is modified to be compatible to the ns-2.27. In addition, it is enhanced to support cooperation in its *forward* method by calling the *sendPacket* method of the *CooperationNode*.

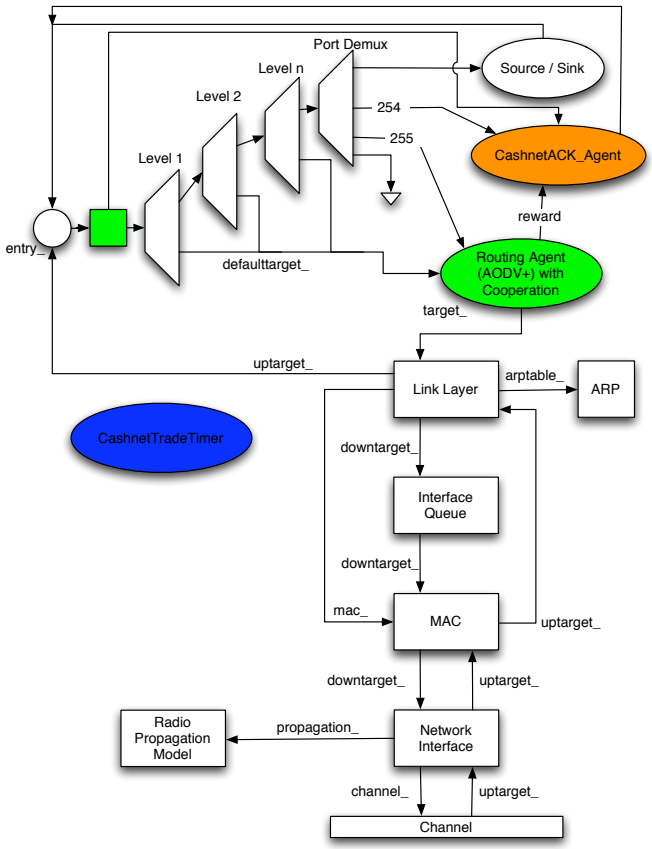


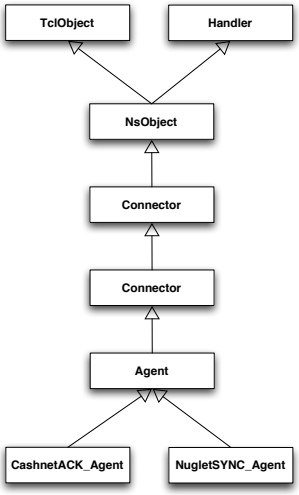
Figure 5.6: Structure of *CashnetNode*

The service-station of the CASHnet scheme is implemented with an additional class called *CashnetServiceStation* (see Fig. 5.7(c)). It contains the location information of the service-station and provides an interface for the creation in the OTcl interpreter (see Listing TCL interface). The trading mechanism of CASHnet, i.e. the purchasing of Traffic Credits with money or Helper Credits, consists of a static list of all service-stations, the method *trade()* of the *CashnetNode* and the timer class *CashnetTradeTimer*(see Fig. 5.7(b)). The *CashnetTradeTimer* calls the *trade* method of the node. *trade* checks if the node is within trade distance of any service-station. If it is, the Helper Credits and real money are traded.

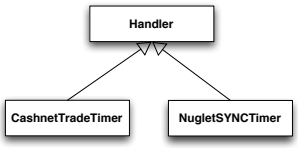
New and Modified Files

aodv/ (Replaced)
The AODV package included in the standard ns2 package is replaced by the AODV+ implementation of [13].

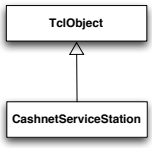
aodv/aodv.h (Modified)
Some modifications are made to make AODV+ compatible with the newest version of ns2



(a) *CashnetACK_Agent* and *NugletSYNC_Agent*



(b) *CashnetTradeTimer* and *NugletSYNCTimer*



(c) *CashnetServiceStation*

Figure 5.7: Class Hierarchies

(2.27).

aodv/aodv.cc (Modified)
The *AODV::forward* method is extended by cooperation parts.

cooperation/cooperationnode.h|**cc** (New)
The *CooperationNode* is a common superclass of *CashnetNode* and *NugletNode* and provides a common interface for them.

cashnet/cashnet.h|**cc** (New)
The *CashnetACK_Agent* and the new packet header for the CASHnet *ACK* message and the corresponding OTcl class bindings provides the ACK message handling for CASHnet.

cashnet/cashnetnode.h|cc (New)
The *CashnetNode* class provides the charging and rewarding mechanisms of CASHnet. The class constructor and certain member variables are bound to OTcl space. That is the way to be able to generate a CashnetNode in OTcl space. In addition, further OTcl shell commands can be defined in *CashnetNode::command* method. Moreover, the *Cashnet-TradeTimer* class provides a timer which is used to periodically start the trading mechanisms.

cashnet/servicestation.h|cc (New)
The *CashnetServiceStation* class models the service station for CASHnet and contains a static list of all *CashnetServiceStation* objects to support the trading process.

common/agent.h|cc (Modified)
The member variables *node_* and *cooperative_* are added and bound to the corresponding OTcl variables. This makes it possible for the agent to find the node, to which it is attached, otherwise there is no possibility to get a reference to the node in C++ because the agents are attached to the node in the OTcl space. This is a common problem in ns2.

common/packet.h (Modified)
A new method is defined to access directly the header of the CASHnet ACK message. *PT_CASHNETACK* is added as new packet type.

trace/cmu-trace.h (Modified)
NO_CASH is defined as new reason for packet drops at nodes with not enough credits to send the packet at the wished destination.

trace/cmu-trace.cc (Modified)
The trace format for the new CASHnet acknowledge message is defined and integrated in the *format* method.

routing/address.h|cc (Modified)
A new method decides whether two addresses are in the same hierarchical subnet or not.

tcl/lib/ns-default.tcl (Modified)
Here the default values of the variables accessible in the OTcl interpreter are set.

tcl/lib/ns-lib.tcl (Modified)
A new cooperation flag is added to the *node-config* interface.

Makefile.in (Modified)
Here the new classes are included in the ns2 make process.

OTcl Interface

In order to keep the configuration of the CASHnet scheme simple, the *node-config* interface is enhanced and all options can be set in the simulation script. The listing 5.2 shows the OTcl interface enhancement for CASHnet.

Listing 5.2: OTcl Interface Extensions for CASHnet

```
$ns_ node-config -cooperation "CASHNET"

$node_(1) set cooperative_ true                ;# Sets the node cooperative (true | false)
$node_(1) set TrafficCredits_ 500              ;# Amount of Traffic Credits
$node_(1) set RealMoneyAccount_ 500            ;# How much real money does the node have?
$node_(1) set HelperCredits_ 0.0               ;# Amount of Helper Credits
$node_(1) set InfinitMoney_ false              ;# Never ending money? (true | false)
$node_(1) set ExchangeRate_ 1                  ;# How many Helper Credits has the node to paid for one Traffic
    Credit?
$node_(1) set ExchangeAmount_ 20.0             ;# How many Traffic Credits are traded at one time?
$node_(1) set TradeThreshold_ 10.0             ;# If the node has less Traffic Credits, it tries to trade.
$node_(1) set TradeDistance_ = 50.0            ;# Within this distance the node can trade Traffic Credits.
$node_(1) set TradeInterval_ = 1.0             ;# Defines the interval between the periodic checks if the node is
    near enough a service station

$ns_ at 1.0 "$node_(1)_trade"                  ;# Starts the trading mechanism of CASHnet
$ns_ at 900.01 "$node_($i)_moneyPaid"   ;# Prints out the amount of Traffic Credits the node paid for sending
    packets
$ns_ at 900.01 "$node_($i)_creditsTraded"      ;# Prints out the amount of Traffic Credits the node traded
$ns_ at 900.01 "$node_($i)_credits"            ;# Prints out the current amount of Traffic Credits
```

5.5 Nuglet Implementation

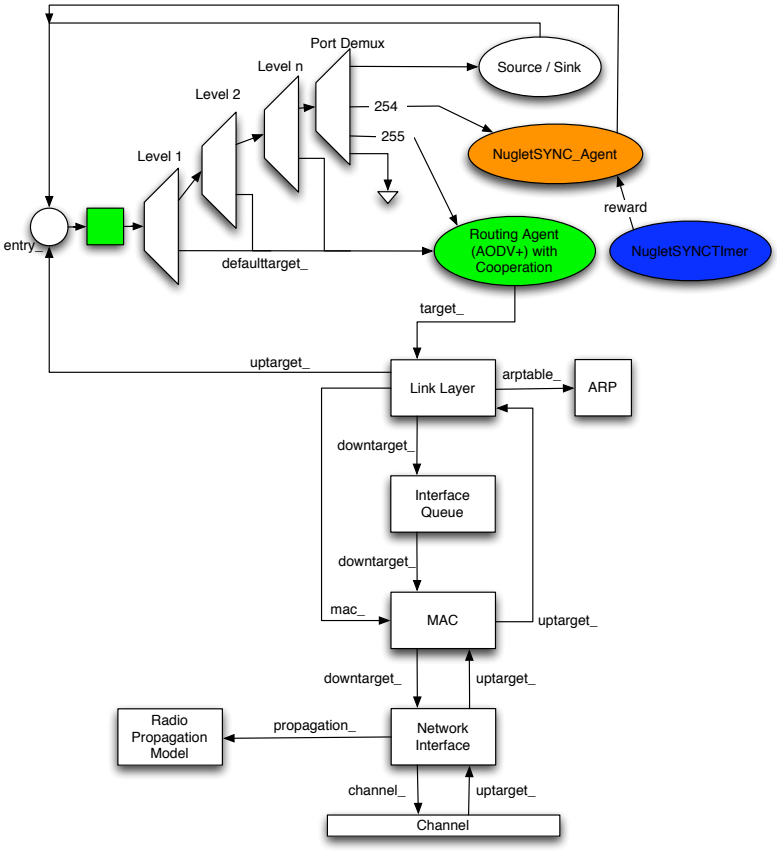


Figure 5.8: Structure of *NugletNode*

The implementation of the Nuglet scheme is similiar to the one of CASHnet. A new subclass of *CooperationNode* is generated (Fig. 5.5). This new *NugletNode* class implements the charging and rewarding mechanism of the Nuglet scheme. Its structure is shown in Fig. 5.8. It contains a *NugletSYNC_Agent* object (Fig. 5.7(a)) which handles the $SYNC$ messages of Nuglet. The cooperation mechanism is integrated in the AODV routing as formerly explained for CASHnet. The *NugletSYNCTimer* periodically starts the delivering of the pending Nuglets counter $pc_{X@N}$ to the corresponding one-hop neighbor per $SYNC$ message.

New and Modified Files

nuglet/nugletnode.h|cc (New)
: The *NugletNode* class provides the charging and rewarding mechanisms of Nuglet. The class is bounded to OTcl similiar to the *CashnetNode*. Further, the *NugletSYNCTimer* periodically calls the *NugletNode::synchronize* method to synchronize the pending counter $pc_{X@N}$ and send the Nuglets to the corresponding one-hop neighbor.

nuglet/nugletsync.h|cc (New)
: The agent *NugletSYNC_Agent* and the new packet for the $SYNC$ message are defined. The agent is responsible for the handling of the $SYNC$ messages.

common/packet.h (Modified)
: A new method is defined to access directly the header of the Nuglet $SYNC$ message. In addition, a new packet type *PT_NUGLETSYNC* is added.

trace/cmu-trace.cc (Modified)
: A new format method for the $SYNC$ is implemented.

Makefile.in (Modified)
: Here the new classes are included in the ns2 make process.

OTcl Interface

The *node-config* interface is also enhanced for the Nuglet scheme and the options of Nuglet can be set in the simulation script. The listing 5.3 presents the OTcl interface enhancement for Nuglet.

Listing 5.3: OTcl Interface Extensions for Nuglet

```
$ns_ node-config -cooperation "NUGLET"

$node_(1) set cooperative_ true                ;# Sets the node cooperative (true | false)
$node_(1) set Nuglets_ 500.0                   ;# Amount of Nuglets
$node_(1) set SYNCInterval_ 1                  ;# Defines the interval between the periodic sending of the SYNC
    messages

$ns_ at 1.0 "$node_(1) sync"                   ;# Starts the synchronizing mechanism of Nuglet
$ns_ at 900.01 "$node_($i) moneyPaid"   ;# Prints out the amount of Nuglets the node paid for sending packets
$ns_ at 900.01 "$node_($i) creditsTraded"      ;# Prints out the amount of Nuglets the node received
$ns_ at 900.01 "$node_($i) credits"            ;# Prints out the current amount of Nuglets
```

Chapter 6

Evaluation

This chapter presents the simulation scenarios and their evaluations. The liveliness of the network for the CASHnet and the Nuglet schemes is measured. The amount and the frequency of starvation of individual nodes are used as the indicator. Furthermore, the packet drops and the resulting overhead are analyzed.

6.1 Simulation Scenarios

The two cooperation schemes CASHnet and Nuglet are evaluated using ns2 and their simplified implementations. For the simulation scenarios only one multi-hop network is considered in order to be compatible to the Nuglet scheme which is actually designed for MANET. Fig. 6.1 presents the basic scenario for the test runs. Different simulation scenarios are made by varying

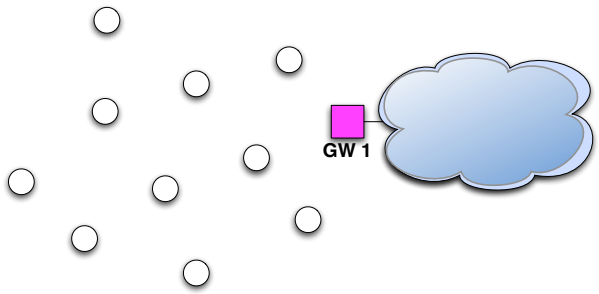


Figure 6.1: Basic evaluation scenario for CASHnet and Nuglet

the amount of service stations and their distribution. Actually, the simulations run with 1, 2, 5, 9 and 12 service stations equally distributed and with 9 or 12 service stations randomly distributed. Fig. 6.2 shows the used simulation scenarios. When the simulation is running with the Nuglet scheme the service stations are removed and the gateway acts as a normal node in the MANET. Furthermore, the packet generation interval at the CBR traffic sources is varied (1, 2, 5, and 10 s). In total 32 (8 x 4) simulation scenarios are investigated and for each of them 20 simulation runs using 20 independent movement files are performed. The simulation script for the CASHnet tests is shown in the listings 6.3. The configurations of the service stations are included from a

Listing 6.1: Example service station file

```
set serviceStation1 [new CashnetServiceStation]
$serviceStation1 set X_ [expr $val(X)/3.0]
$serviceStation1 set Y_ [expr $val(X)/2.0]
$serviceStation1 set Z_ 0.0

set serviceStation2 [new CashnetServiceStation]
$serviceStation2 set X_ [expr 2*$val(X)/3.0]
$serviceStation2 set Y_ [expr $val(X)/2.0]
$serviceStation2 set Z_ 0.0
```

Listing 6.2: Example lines of a node movement file

```
#
# nodes: 40, speed type: 1, min speed: 1.00, max speed: 10.00
# avg speed: 3.67, pause type: 2, pause: 10.00, max x: 1500.00, max y: 800.00
#
$node_(0) set X_ 1339.865501781931
$node_(0) set Y_ 523.560828552733
$node_(0) set Z_ 0.000000000000
...
$ns_ at 0.000000000000 "$node_(0) setdest 1347.356779559099 102.346239166338 3.020117394902"
$ns_ at 0.000000000000 "$node_(1) setdest 603.549904082525 22.785051897934 7.340209978810"
...
$god_ set-dist 0 1 4
$god_ set-dist 0 2 3
...
$god_ set-dist 38 39 1
$ns_ at 0.030803485331 "$god_ set-dist 18 25 1"
...
$ns_ at 223.791821804280 "$god_ set-dist 15 36 3"
$ns_ at 223.797528851594 "$node_(8) setdest 903.090542727809 531.433502803131 0.000000000000"
$ns_ at 224.101320059278 "$god_ set-dist 10 14 2"
$ns_ at 224.101320059278 "$god_ set-dist 10 30 1"
...
ns_ at 899.839645853655 "$god_ set-dist 18 23 2"
$ns_ at 899.839645853655 "$god_ set-dist 18 28 3"
#
# Destination Unreachables: 1829
#
# Route Changes: 37291
#
# Link Changes: 3491
#
# Node | Route Changes | Link Changes
#    0 |          2073 |          120
#    1 |          1839 |          146
...
#   39 |          2106 |          166
```

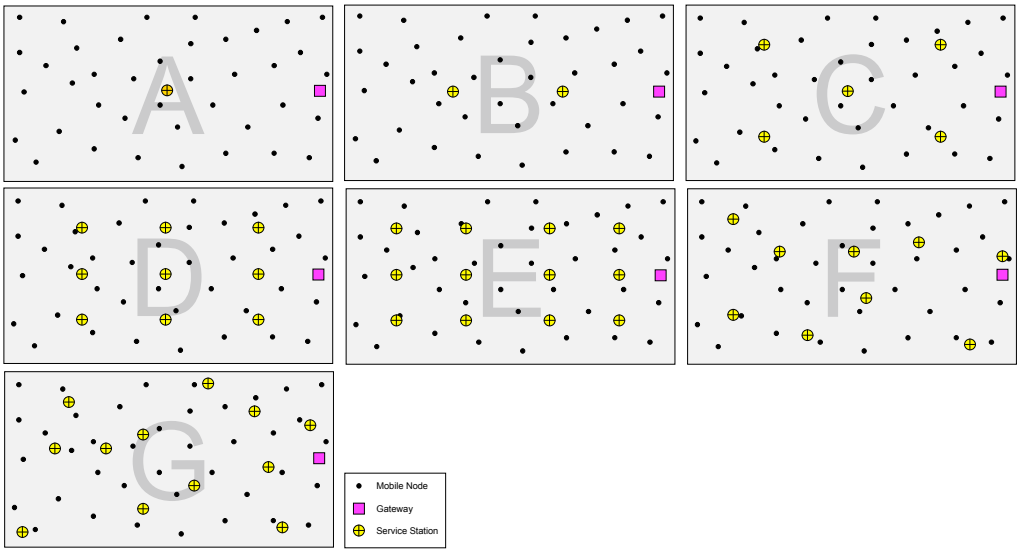


Figure 6.2: Simulation scenarios with distribution of the service stations

separate file (see listing 6.1). Furthermore, the movements of the nodes are also included from an external file (Listing 6.2). This mobility file is generated using the *setdest* program of the ns2 package and provides movements according to the Random Waypoint Mobility Model. The movement and service station files as well as the handing-over of parameters enable the reuse of the same simulation script for all of our CASHnet scenarios. The simulation script for Nuglet is corresponding to the one of CASHnet. The table 6.1 shows the parameters for both schemes.

Parameter	Value	
	Nuglet	CASHnet
Initial balance of virtual money account	100 Nuglets	100 Traffic Credits
Initial balance of real money account	–	500
Nuglet synchronization interval	5s	–
Traffic / Helper Credits exchange rate	–	1:1
Exchange threshold at service stations	–	10 Helper Credits
Distance threshold to service stations	–	50 m
Number of service stations	–	1, 2, 5, 9 or 12
Simulation area	1500 m x 800 m	
Number of mobile nodes	40	
Transmission range	250 m	
Mobility model	random mobility	
Node speed	uniformly distributed between 1 and 10 m/s	
Pause time	uniformly distributed between 0 and 20 s	
Packet generation interval	1, 2, 5, 10 s	
Routing	AODV+	
Simulation time	900 s	

Table 6.1: Simulation parameters

Listing 6.3: Simulation script for CASHnet

```
proc getopt {argc argv} {
    global opt
    lappend optlist movs file servicestations nnl nnr packetinterval
    for {set i 0 } { $i < $argc } {incr i } {
        set arg [lindex $argv $i]
        if {[string range $arg 0 0] != "-"} continue
        set name [string range $arg 1 end]
        set opt($name) [lindex $argv [expr $i+1]]
    }
}

proc stop {} {
    global ns_ tracefd
    $ns_ flush-trace
    close $tracefd
}

# Read command line args
getopt $argc $argv
# ======================================================================
# Define options
# ======================================================================
set val(prop)           Propagation/TwoRayGround   ;# radio-propagation model
set val(netif)          Phy/WirelessPhy            ;# network interface type
set val(mac)            Mac/802_11                 ;# MAC type
set val(ifq)            Queue/DropTail/PriQueue    ;# interface queue type
set val(ll)             LL                         ;# link layer type
set val(ant)            Antenna/OmniAntenna        ;# antenna model
set val(ifqlen)         50                         ;# max packet in ifq
set val(nnl)            $opt(nnl)                  ;# number of mobilenodes in left area
set val(nnr)            $opt(nnr)                  ;# number of mobilenodes in right area
set val(bsn)            2                          ;# number of basestations
set val(rp)             AODV                       ;# routing protocol
set val(TrafficCredits) 100.0                      ;# start credit for all nodes
set val(ExchangeRate)   1.0                        ;# exchange rate helper points / traffic points
set val(RealMoneyAccount) 500.0
set val(X)              1500
set val(Y)              800

set ns_         [new Simulator]
$ns_ use-newtrace

set tracefd     [open $opt(file) w]
$ns_ trace-all $tracefd

set topo       [new Topography]
$topo load_flatgrid [expr 2 * $val(X)] $val(Y)            ;# create topology

set god_ [create-god [expr $val(nnl) + $val(nnr) + $val(bsn) + 1]]

#---------Addressing----------#
$ns_ node-config -addressType hierarchical
AddrParams set domain_num_ 3
AddrParams set cluster_num_ {1 1 1}
AddrParams set nodes_num_ {1 41 41}
Agent/AODV set gw_discovery 1

set router1 [$ns_ node 0.0.0]

$ns_ node-config -adhocRouting $val(rp) \
                 -llType $val(ll) \
                 -macType $val(mac) \
                 -ifqType $val(ifq) \
                 -ifqLen $val(ifqlen) \
                 -antType $val(ant) \
                 -propType $val(prop) \
                 -phyType $val(netif) \
                 -channel [new Channel/WirelessChannel] \
                 -topoInstance $topo \
                 -agentTrace ON \
                 -routerTrace ON \
                 -macTrace OFF \
                 -movementTrace OFF \
                 -wiredRouting ON \
                 -cooperation CASHNET

set gw1 [$ns_ node 1.0.0]
set gw2 [$ns_ node 2.0.0]

$ns_ duplex-link $gw1 $router1 100Mb 2ms DropTail
$ns_ duplex-link $router1 $gw2 100Mb 2ms DropTail
```

```
# Creation of the mobile nodes
$ns_ node-config -wiredRouting OFF

# Create mobile nodes for sector 1
for {set i 0} {$i < $val(nnl)} {incr i} {
    set node_($i) [$ns_ node 1.0.[expr $i+1]]
    #puts "\nNode $i $node_($i) [$node_($i) node-addr]"
    $node_($i) random-motion 0
    $node_($i) base-station [AddrParams addr2id [$gw1 node-addr]]
    $node_($i) set cooperative_ true
    $node_($i) set TrafficCredits_ $val(TrafficCredits)
    $node_($i) set ExchangeRate_ $val(ExchangeRate)
    $node_($i) set RealMoneyAccount_ $val(RealMoneyAccount)
}
# Create mobile nodes for sector 2
for {set i 0} {$i < $val(nnr)} {incr i} {
    set node_([expr $i + $val(nnl)]) [$ns_ node 2.0.[expr $i+1]]
    $node_([expr $i + $val(nnl)]) random-motion 0
    $node_([expr $i + $val(nnl)]) base-station [AddrParams addr2id [$gw2 node-addr]]
    $node_([expr $i + $val(nnl)]) set cooperative_ true
    $node_([expr $i + $val(nnl)]) set TrafficCredits_ $val(TrafficCredits)
    $node_([expr $i + $val(nnl)]) set ExchangeRate_ $val(ExchangeRate)
    $node_([expr $i + $val(nnl)]) set RealMoneyAccount_ $val(RealMoneyAccount)
}

# positions for gateways, router
$gw1 set X_ 1490.0
$gw1 set Y_ 400.0
$gw1 set Z_ 0.0

$gw2 set X_ 1500.0
$gw2 set Y_ 400.0
$gw2 set Z_ 0.0

$router1 set X_ 1495.0
$router1 set Y_ 400.0
$router1 set Z_ 0.0

# Get the servicestations
source $opt(servicestations)

#Create connections nodes of sector 1 -> gw2
for {set i 0} {$i < $val(nnl)} {incr i} {
    set src_($i) [new Agent/UDP]
    set dst_($i) [new Agent/Null]
    $ns_ attach-agent $node_($i) $src_($i)
    $ns_ attach-agent $gw2 $dst_($i)
    $ns_ connect $src_($i) $dst_($i)
    $src_($i) set fid_ 1
    set cbr_($i) [new Application/Traffic/CBR]
    $cbr_($i) set packetSize_ 512
    $cbr_($i) set interval_ $opt(packetinterval)
    $cbr_($i) attach-agent $src_($i)
    $ns_ at 1.01 "$cbr_($i) start"
    $ns_ at 850.0 "$cbr_($i) stop"
}
for {set i 0} {$i < $val(nnl) + $val(nnr)} {incr i} {
    $ns_ at 1.0 "$node_($i) trade"
}
# Import movement patterns
source $opt(movs)

for {set i 0} {$i < $val(nnl) + $val(nnr) } {incr i} {
    $ns_ at 900.0 "$node_($i) reset";
    $ns_ at 900.01 "$node_($i) moneyPaid"
    $ns_ at 900.01 "$node_($i) creditsTraded"
    $ns_ at 900.01 "$node_($i) credits"
}
$ns_ at 900.0 "$gw1 reset";
$ns_ at 900.0 "$gw2 reset";
$ns_ at 900.001 "stop"
$ns_ at 900.01 "$ns_ halt"
$ns_ run
```

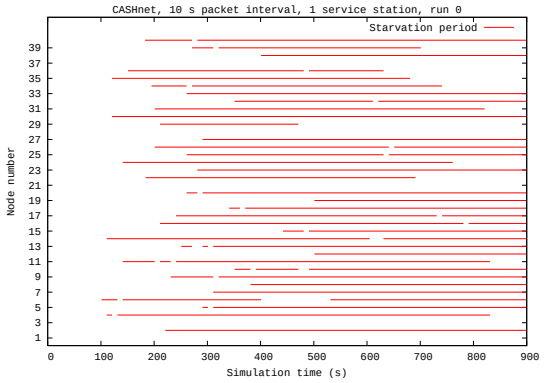
6.2 Results

The discussion starts with the starvation properties of the CASHnet and the Nuglet scheme. Afterwards, the overall protocol performance and the protocol overhead are analyzed. The mean results over the 20 simulation runs from 24 scenarios are combined by the Fig. 6.7, 6.9, 6.10, 6.11, and 6.12. Each x-axis label consists of two lines. The first line codes the number of service stations (1, 2, 5, 9, 12 for CASHnet and 0 for Nuglet). The used service station distributions are A, B, C, D, and E as presented in Fig. 6.2 on page 49. The second line indicates the packet generation interval (1, 2, 5, 10 s). In the figures the different packet generation intervals are separated by vertical lines.

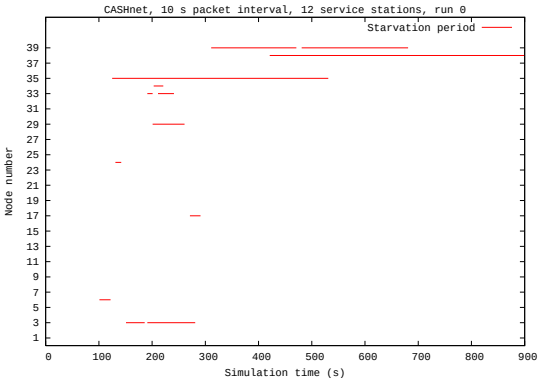
6.2.1 Starvation

The starvation describes the node's inability to send self-generated packets due to the lack of virtual money (Traffic Credits or Nuglets). Fig. 6.3 presents the starvation periods of each node during a single simulation run for 3 scenarios with a packet generation interval of 10 s. The red lines indicates the starvation periods of the nodes. Because the nodes frequently cross the area of a service station in scenario E (Fig. 6.2 on page 49) and can trade Traffic Credits, the starvation periods of the nodes are almost eliminated. The Nuglet scenario performs better as the scenario A, because with only one service station in the CASHnet scenario the probability of being able to trade credits is low and the rewarding mechanism of Nuglet is preferred. The behavior of the first ten nodes of scenario E is additionally displayed in spatial way in Fig. 6.8. The transmission range of the gateway is marked as a semicircle as well as the service stations are shown as circles. The nodes follow their paths beginning at the arrow. The starvation events are indicated by a cross with label with the format <B (begin) | E (end)><time><Traffic Credits><Helper Credits><real money>. The histograms in Fig. 6.5 sort the starvation events according to their durations. A box represents the mean number of starvation events in this duration category for the 20 simulation runs. Because a large number of long starvations is normally much worse than many small starvations, the longer the starvation duration is (x-axis), the smaller the number of starvations should be (y-axis). The average number of nodes which are starving for quite the complete simulation time is rather low for all scenarios. The scenarios F and G with the randomly distributed service stations perform worse than their correspondent scenarios D and E with aligned service stations. Long starvation periods occur more frequently for them. Thus the equal distribution of the service stations raises their positive influence on the overall network liveliness. Further, the effect of additional service stations flattens out with the number of them, i.e. the raise from 5 to 9 service stations brings more benefits than the raise from 9 to 12.

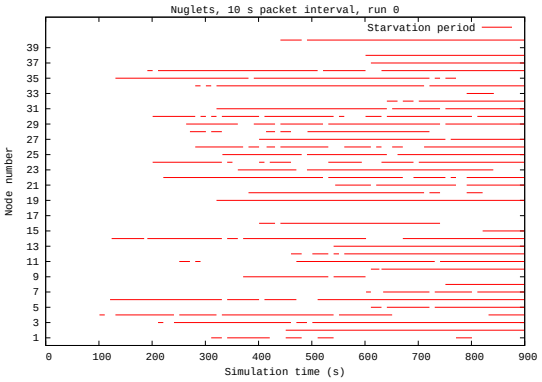
The histograms in Fig. 6.4 present the results for the scenarios B and E. The increasing of the number of service stations reduces the number of starvations and shortens the single starvation periods (see also Fig. 6.5). By increasing the sending rate in a scenario, e.g. E, with a high number of service stations, the number of starvation periods becomes higher. The high number of sent packets causes the nodes to run out of Traffic Credits faster. The probability of starving nodes grows. Due to the high number of service stations especially the amount of short starvations goes up. The nodes more often meet a service station and are more frequently able



(a) Scenario A CASHnet



(b) Scenario E CASHnet

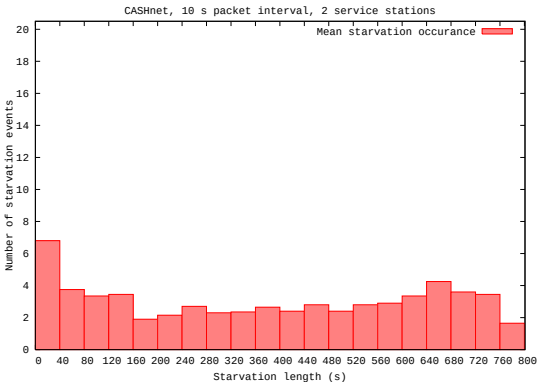


(c) Scenario Nuglet

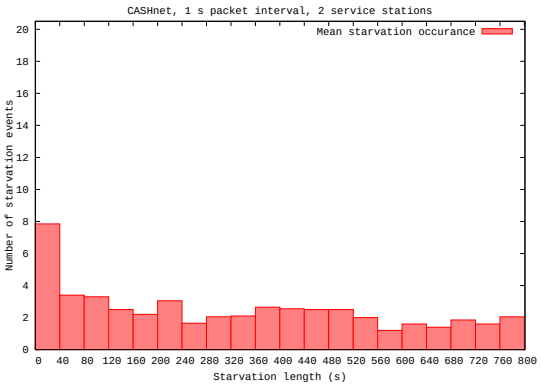
Figure 6.3: Starvation periods for all nodes during a single simulation run

to trade credits than with a low density of service stations. For a scenario with a low number of service stations, e.g. B, the increasing of the packet generation rate has a different effect. The long starvation periods are mainly transformed in short ones. The nodes receive more Helper Credits because of the increased traffic and so can exchange more of them at the service station.

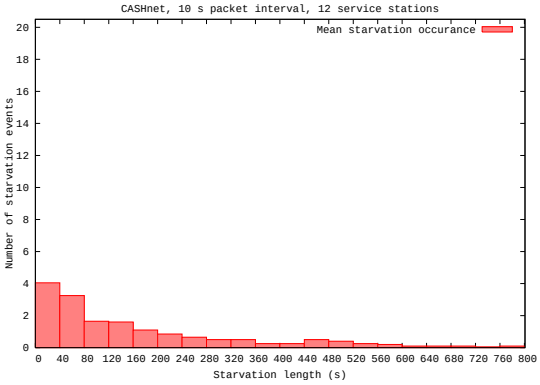
Fig. 6.7 illustrates the average starvation length for a node in CASHnet and Nuglet with a total simulation time of 900 s. The two schemes CASHnet and Nuglet perform poorly under high network load. CASHnet is generally better than Nuglet, if there are at least five service stations deployed. The CASHnet scheme performs worse with a low number of service stations because the rewarding overhead is higher than for the Nuglet scheme (see section 6.2.3) and moreover the healthy effect of trading credits is small with few service stations. If it is looked at the results under low network load, CASHnet performs quite well in contrast to Nuglet. At first sight, it is surprising that the values for scenario B with two service stations are worse than for



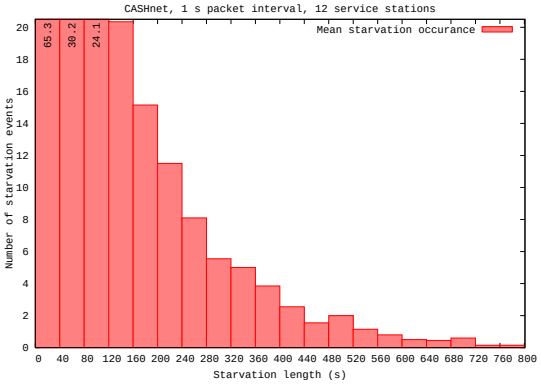
(a) Scenario B 0.1 packet/s



(b) Scenario B 1 packet/s



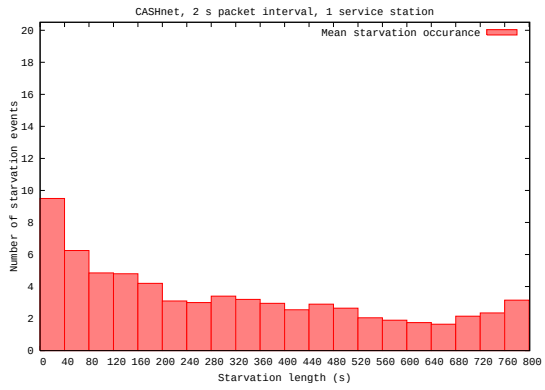
(c) Scenario E 0.1 packet/s



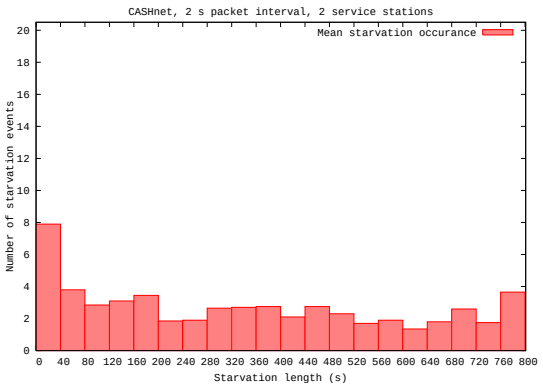
(d) Scenario E 1 packet/s

Figure 6.4: Comparison of two CASHnet scenarios with two different packet generation intervals

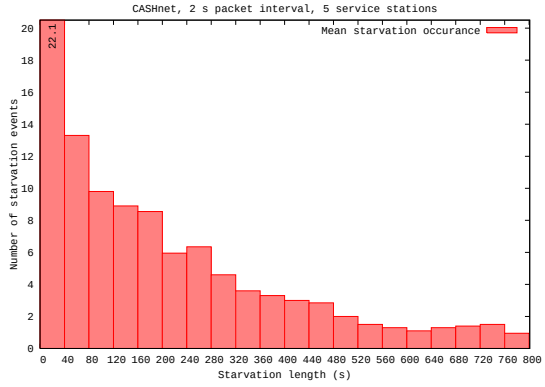
scenario A with one service station. The same is valid for the goodput as shown later in Fig. 6.9. This could be caused by the Random Waypoint Mobility Model. It leads in node paths which have a higher probability to go through the center of the simulation area. The nodes do not pass at a service station frequently and therefore can not refill their Traffic Credits accounts.



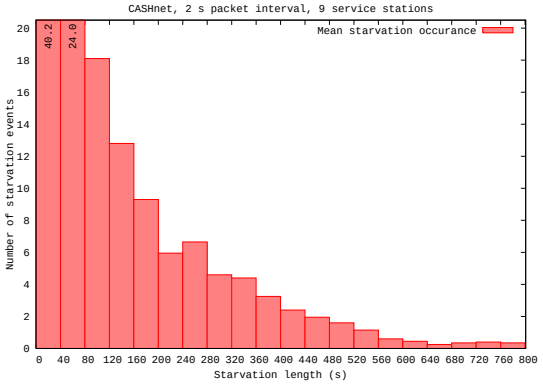
(a) Scenario A CASHnet



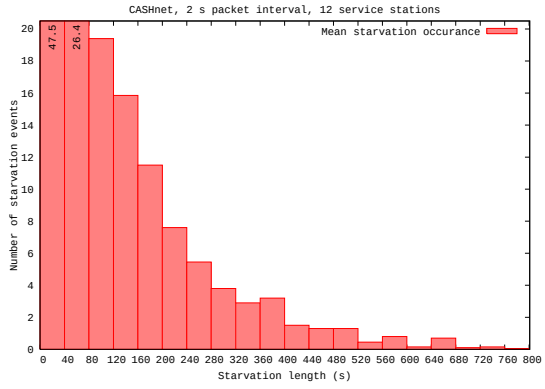
(b) Scenario B CASHnet



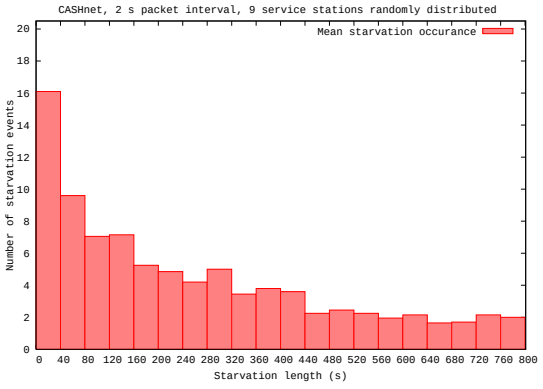
(c) Scenario C CASHnet



(d) Scenario D CASHnet

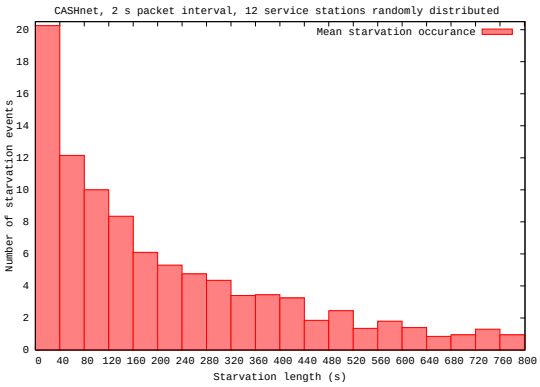


(e) Scenario E CASHnet

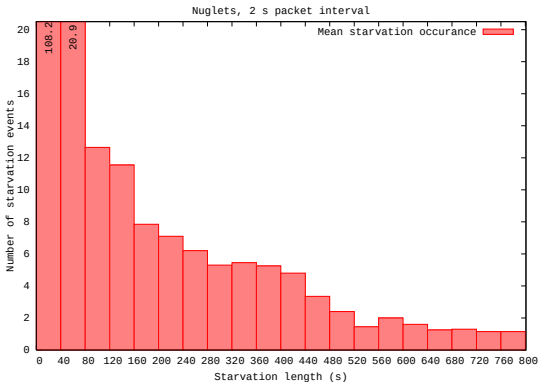


(f) Scenario F CASHnet

Figure 6.5: Mean number of starvation events per duration category with 2 s packet generation interval



(g) Scenario *G* CASHnet



(h) Scenario Nuglet

Figure 6.6: Mean number of starvation events per duration category with 2 s packet generation interval

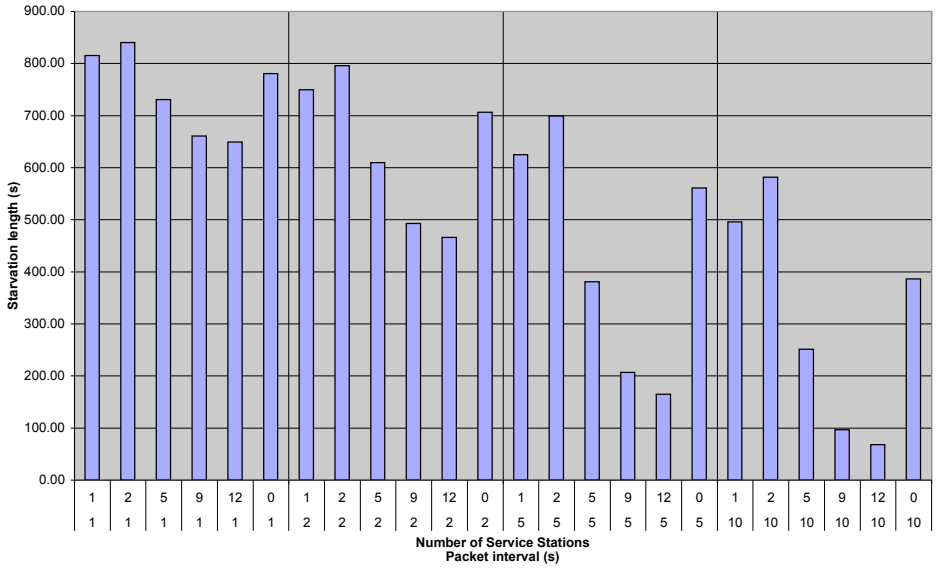


Figure 6.7: Average starvation length for a node in CASHnet and Nuglet

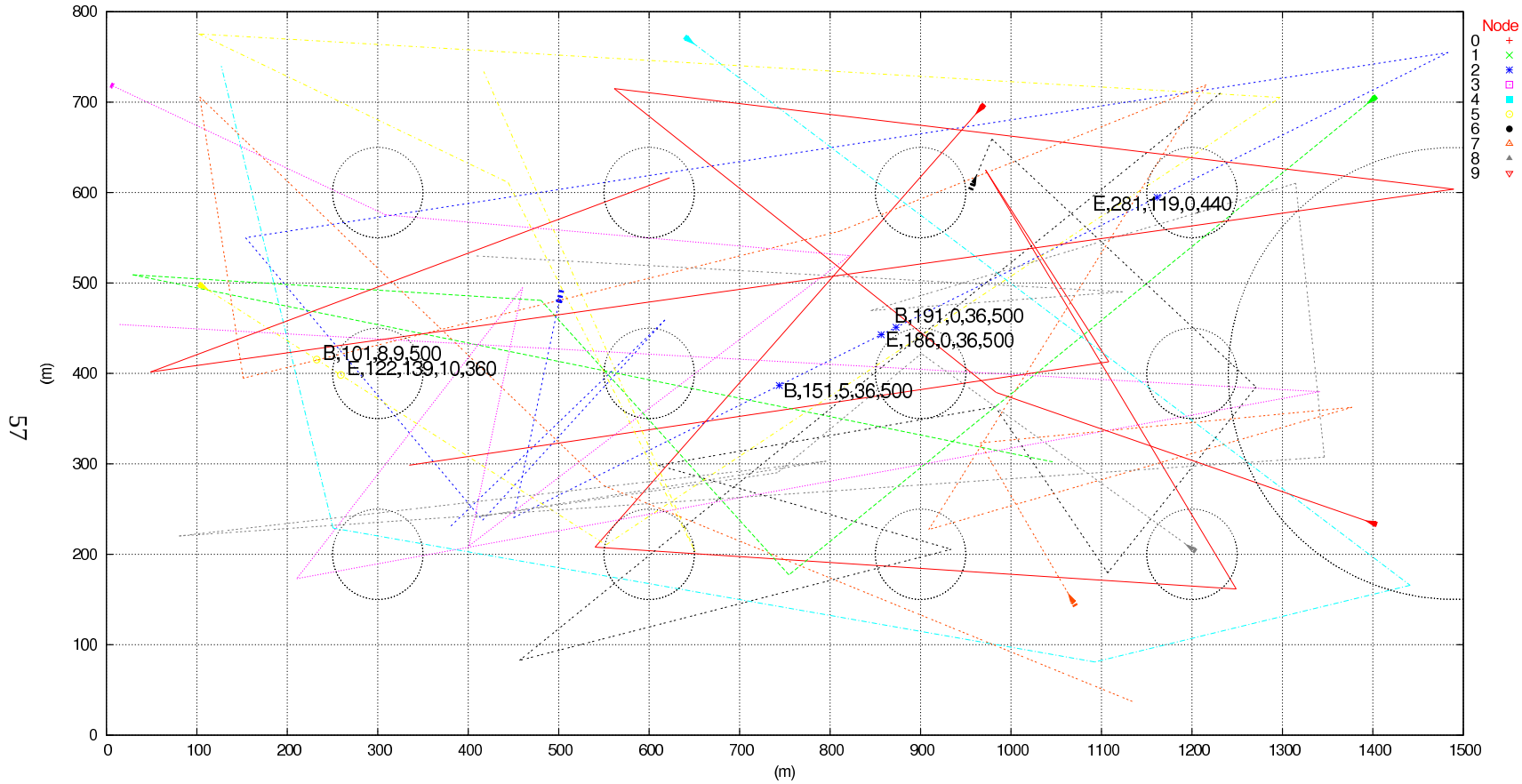


Figure 6.8: CASHnet, 10s packet interval, 12 service stations

6.2.2 Goodput and Packet Drops

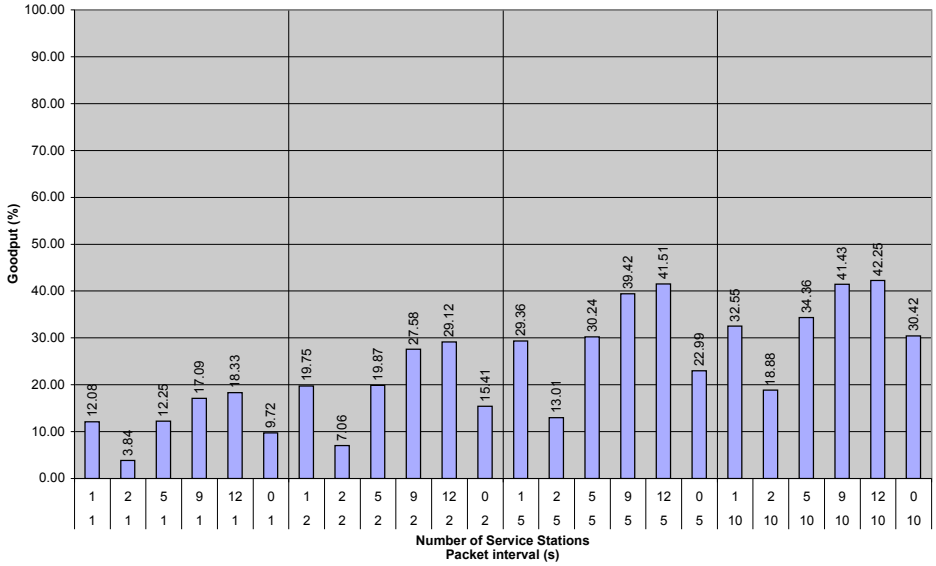


Figure 6.9: Goodput in % for CASHnet and Nuglet

The goodput is defined as the percentage the number of received packets have in the number of sent packets (see Fig. 6.9). It is very low either for the CASHnet scheme or for Nuglet scheme. But it is very promising that CASHnet performs remarkable better than Nuglet if the packet generation rate is high. The goodput for the scenario with 12 service stations is almost twice the goodput for Nuglet when the packet generation interval is 1 or 2 s.

Fig. 6.10 shows the transmitted and the dropped packets. The packets are categorized in packets received by their destination, packets dropped because of lack of virtual money (Traffic Credits or Nuglets), and packets dropped for other reasons. The drop reasons will be analyzed later. The number of received packets increases as expected when increasing the number of service stations. The abnormal behavior of the scenario B with 2 service stations deployed is again an effect of the Random Mobility Model. At the same time, the number of packets dropped for other reasons increases because of more packets sent to the network, i.e. packet transmission are not prevented by lack of virtual money.

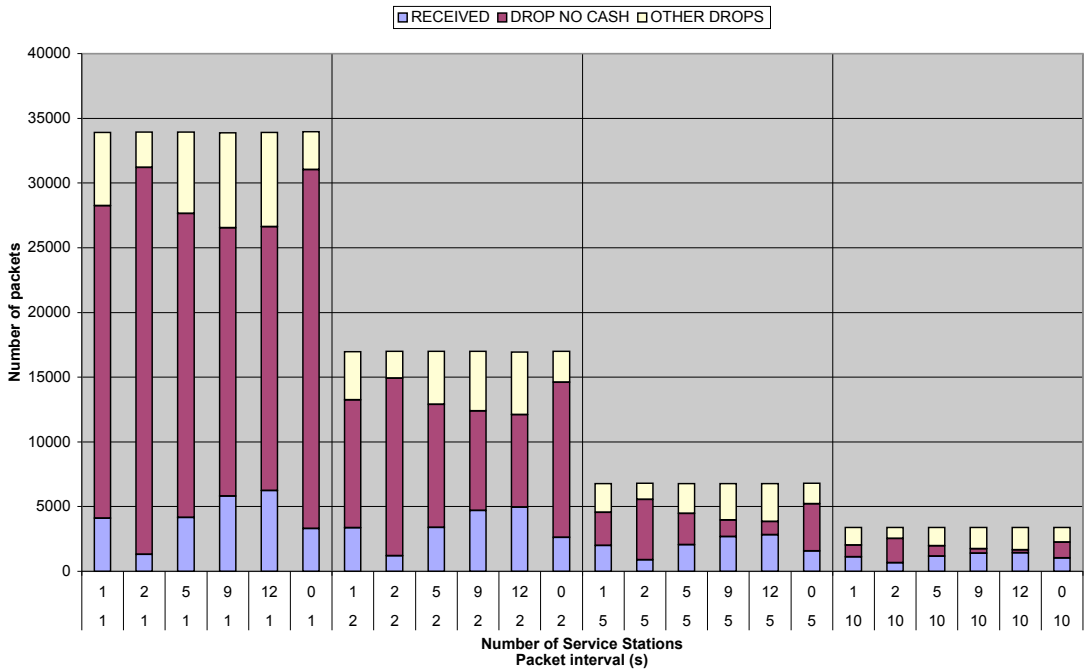


Figure 6.10: Packet sent destinations for CASHnet and Nuglet

The packet drops are classified in Fig. 6.11. By analyzing the trace file the following packet drop reasons are found:

NO CASH lack of virtual money

NO ROUTE no available route to destination

LOOP routing loop

CALLBACK MAC layer callback timer

ARP delay in ARP

The $LOOP$ and ARP drop reasons turn up very rarely. They do not vary very much between all the scenarios. The occurrences of $NO\ CASH$, $NO\ ROUTE$ and $CALLBACK$ are considerable. The main reason for packet drops in the Nuglet scheme is $NO\ CASH$. This is an indication that under high traffic it is very difficult to generate enough traffic to build up a self-perpetuating cycle. The node does not always receive enough Nuglets to send all its self-generated packets. The ability to buy credits in CASHnet reduces this starvation problem. It offers the node the possibility to buy its right to transmit its own packets.

6.2.3 Cooperation Protocol Overhead

The protocol overhead of both schemes is illustrated in Fig. 6.12. The overhead introduced by the ACK packets in the CASHnet scheme is much higher than the one caused by the $SYNC$

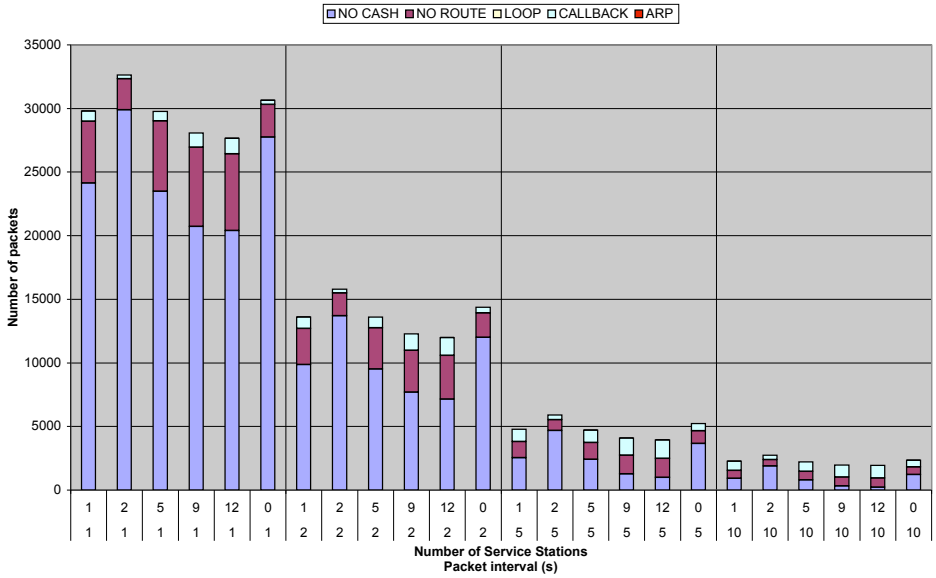


Figure 6.11: Packet drop reasons for CASHnet and Nuglet

packets of Nuglet. In CASHnet each successful forwarding of a packet is rewarded by a separate ACK packet, whereas in the Nuglet scheme the rewards are collected and periodically sent with a $SYNC$ packet to the reachable nodes. The higher rewarding cost of CASHnet can be the reason for more packet drop events from the lower layers.

There exists an additional overhead for both cooperation schemes that is not included in the simulation because the security concepts are left aside for the implementation in ns2. The packets in CASHnet and Nuglet contain additional headers for authentication and data integrity. Furthermore, the security associations in the Nuglet scheme have to be built up, as well as the paths in the CASHnet scheme must be authenticated. This causes additional control packets and increases the overall protocol overhead. There are also added burdens for a node in the form of used processing power for the security concept of both schemes.

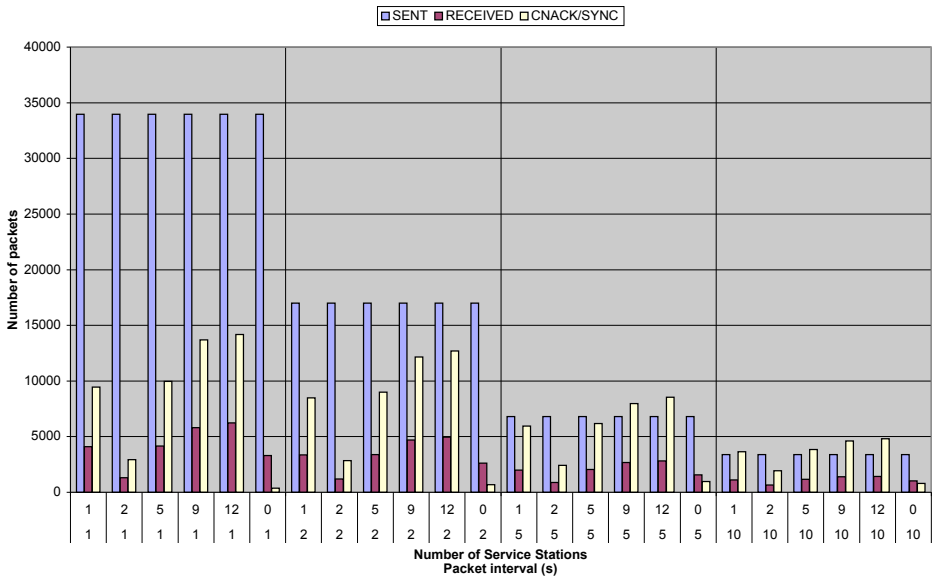


Figure 6.12: Overhead for CASHnet and Nuglet

Chapter 7

Conclusions and Outlook

7.1 Conclusions

The cooperation scheme CASHnet offers a highly decentralized accounting and security architecture for multi-hop cellular networks. It is a cooperation scheme based on rewards. Selfish nodes are not forbidden, but they are encouraged to participate in packet forwarding by the mean of rewards. The CASHnet scheme keeps the freedom of choice at the node's side. The CASHnet scheme combines sender and receiver based payment and does not charge pure ad-hoc traffic.

The evaluation with the Nuglet scheme shows that it is very difficult to achieve the equilibrium between the sending of own packets and the forwarding of packet for the benefit of others. This fact confirms the decision of separating the right for transmission of self-generated packet from the forwarding activity for other nodes. The buyable right for transmission in the CASHnet cooperation scheme generally reduces the starvation of the nodes if enough service-stations are available. Therefore the CASHnet scheme performs better for most of the investigated criteria. The rather high protocol overhead is one of the drawbacks of CASHnet. Furthermore, the number and the distribution of the service stations play an important role for the performance of the cooperation scheme. The evaluations show that a regular distribution of about 9 or 12 service stations (see scenario D and E in Fig. 6.2 on page 49) gives back the best results. Moreover, the CASHnet scheme performs significantly better under high network load than the Nuglet scheme.

The results of the simulations highly depend on the used mobility model as the values for 2 service stations show.

In summary it may be said that the CASHnet scheme provides a promising approach to stimulate cooperation in a multi-hop cellular network. Based on this approach more investigations may be done in the future.

7.2 Outlook

There is a lot of work which could not be done in this thesis. Therefore, future work on the topic of CASHnet will include the following topics:

- The granularity of the charging and rewarding mechanism may be fine-tuned.

- An implementation of the security concept of CASHnet and Nuglet has to be done. This will serve as an evaluation of the overall cooperation protocol overhead.
- The movements of the nodes can be modeled by using different mobility models. This shows the effect of the mobility model on the results of the simulation. As seen in this thesis the mobility model can have an impact on the results.
- A more realistic model of the node movements can include an automatic movement of the node towards the next service stations as soon as it begins to starve.
- Simulations with real world mobility data, e.g. from a mobile telephone carrier, can also contribute to more realistic scenarios which can verify the now received results.
- A probability function inside the cooperation node can model the willingness of the node to cooperate. The Nuglet and the CASHnet scheme can be evaluated considering the effect of non-cooperative nodes.
- The effect of a more changing network with nodes leaving and joining the network can be investigated.
- The CASHnet scheme has to be compared with other incentive cooperation mechanisms.
- A mechanism to prevent a cooperation proxy in the multi-hop cellular network has to be integrated in the CASHnet scheme. A cooperation proxy exploits the non-charging of pure ad-hoc traffic in the current CASHnet scheme by accepting ad-hoc traffic and forwarding it within one-hop distance to the gateway. One possibility is to charge for ad-hoc traffic, too.
- A payment for MANET internal traffics may be introduced. This prevents also the problems with a cooperation proxy.
- A solution for receiver nodes with no Traffic Credits has also to be found. A node with an empty Traffic Credits account can not receive packet for which it is the destination. If this situation occurs the gateways should be informed and not forward further packets in the receiver network to this destination node. A signaling mechanism has to be defined.
- Full trusted mobiles nodes can act as mobile service stations for the provider. This allows the provider to keep the cost for deploying static service stations low.
- The security mechanism of CASHnet has to be implemented in a real system. As a tamper resistant device a Java smart-card can be used. It will manage the protected protocol functions with a Java Card 2.0 applet.
- As last step, a complete implementation of the CASHnet scheme including the security mechanism can be done. Then, experiments with two multi-hop cellular networks may be performed.

Bibliography

[1] T. Camp, J. Boleng, and V. Davies, "A survey of mobility models for ad hoc network research," *Wireless Communications & Mobile Computing (WCMC): Special issue on Mobile Ad Hoc Networking: Research, Trends and Applications*, vol. 2, no. 5, pp. 483–502, 2002.

[2] H.-Y. Hsieh and R. Sivakumar, "On using the ad-hoc network model in cellular packet data networks," in *Proceedings of ACM International Symposium on Mobile Ad Hoc Networking and Computing (MobiHoc)*, 2002.

[3] M. Mauve, J. Widmer, and H. Hartenstein, "A survey on position-based routing in mobile ad-hoc networks," *IEEE Network*, vol. 15, no. 6, pp. 30–39, November 2001.

[4] S. Basagni, I. Chlamatac, V. Szrotiuk, and B. Woodward, "A distance routing effect algorithm for mobility (dream)," in *Proceedings of the ACM/IEEE International Conference on Mobile Computing and Networking (MOBICOM)*, Dallas, TX, USA, 1998, pp. 76–84.

[5] E. Royer and C. Toh, "A review of current routing protocols for ad-hoc mobile wireless networks," 1999.

[6] C. Perkins and P. Bhagwat, "Highly dynamic destination-sequenced distance-vector routing (DSDV) for mobile computers," in *ACM SIGCOMM'94 Conference on Communications Architectures, Protocols and Applications*, London, UK, August 1994, pp. 234–244.

[7] P. Jacquet, P. Muehlethaler, T. Clausen, A. Laouiti, A. Qayyum, and L. Viennot, "Optimized link state routing protocol for ad hoc networks," in *Proceedings of IEEE International Multitopic Conference (INMIC)*, Lahore, Pakistan, December 2001.

[8] T.Clausen and P. Jacquet, "Optimized link state routing protocol," IETF RFC 3626, October 2003. [Online]. Available: http://www.ietf.org/rfc/rfc3626

[9] V. D. Park and M. S. Corson, "A highly adaptive distributed routing algorithm for mobile wireless networks," in *Proceedings of IEEE INFOCOM*, April 1997, pp. 1405–1413.

[10] D. Johnson, D. Maltz, and J. Broch, "Dsr the dynamic source routing protocol for multihop wireless ad hoc networks," 2001.

[11] C. Perkins, "Ad hoc on demand distance vector (aodv) routing," January 2002.

[12] C.Perkins, E. Belding-Royer, and S. Das, “Ad hoc on-demand distance vector (aodv) routing,” IETF RFC 3561, July 2003. [Online]. Available: http://www.ietf.org/rfc/rfc3561

[13] A. Hamidian, “A study of internet connectivity for mobile ad hoc networks in ns 2,” Master’s thesis, Departement of Communication Systems, Lund Institute of Technology, Lund University, Sweden, January 2003.

[14] R. Wakikawa, J. Malinen, C. Perkins, A. Nilsson, and A. Tuominen, “Global connectivity for ipv6 mobile ad hoc networks,” IETF Internet Draft, November 2001.

[15] S. McCanne and S. Floyd. (2004, September) ns2 network simulator 2. [Online]. Available: http://www.isi.edu/nsnam/ns

[16] A. Weyland and T. Braun, “Cashnet - cooperation and accounting strategy for hybrid networks,” in *Proceedings of Workshop on Modeling and Optimization in Mobile, Ad Hoc and Wireless Networks (WiOpt)*, Cambrigde, UK, March 2004.

[17] ——, “Cooperation and accounting strategy for multi-hop cellular networks,” in *Proceedings of IEEE Workshop on Local and Metropolitan Area Networks (LANMAN)*, Mill Valley, CA, USA, April 2004.

[18] A. Weyland, T. Staub, and T. Braun, “Liveliness evaluation of a cooperation and accounting strategy in hybrid network,” in *Proceedings of Workshop on Applications and Services in Wireless Networks (ASWN)*, Boston, MA, USA, August 2004.

[19] L. Buttyán and J.-P. Hubaux, “Stimulating cooperation in self-organizing mobile ad hoc networks,” *ACM Mobile Networks & Applications*, vol. 8, no. 5, Oct. 2003.

[20] S. Buchegger and J.-Y. L. Boudec, “Performance analysis of the confidant protocol (cooperation of nodes - fairness in dynamic ad-hoc networks),” in *Proceedings of ACM International Symposium on Mobile Ad Hoc Networking and Computing (MobiHoc)*, Lausanne, Switzerland, June 2002.

[21] M. Jakobsson, J.-P. Hubaux, and L. Buttyán, “A micro-payment scheme encouraging collaboration in multi-hop cellular networks,” in *Proceedings of International Financial Cryptography Conference*, Gosier, Guadeloupe, Jan. 2003.

[22] S. Zhong, J. Chen, and Y. R. Yang, “Sprite: A simple, cheat-proof, credit-based system for mobile ad-hoc networks,” in *Proceedings of IEEE INFOCOM*, San Francisco, CA, USA, March-April 2003.

[23] B. Lamparter, K. Paul, and D. Westhoff, “Charging support for ad hoc stub networks,” *Elsevier J. Computer Communications*, vol. 26, no. 13, pp. 1504–1514, Aug. 2003.

[24] N. B. Salem, L. Buttyán, J.-P. Hubaux, and M. Jakobsson, “A charging and rewarding scheme for packet forwarding in multi-hop cellular networks,” in *Proceedings of ACM International Symposium on Mobile Ad Hoc Networking and Computing (MobiHoc)*, Annapolis, MD, USA, June 2003.

[25] IBM, “Ibm 4758 pci cryptographic coprocessor,” June 1997.

[26] K. Fall, K. Varadhan, and the VINT project. (2004, September) The ns manual. [Online]. Available: http://www.isi.edu/nsnam/ns/ns-documentation.html

[27] R. Griswold. (2004, September) Ns-2 trace formats. [Online]. Available: http://www.k-lug.org/~griswold/NS2/ns2-trace-formats.html

[28] J. Yoon, M. Liu, and B. Noble, “Random waypoint considered harmful,” in *Proceedings of IEEE INFOCOM*, March - April 2003.