New Implementations into Simulation Software NS-2 for Routing in Wireless Ad-Hoc Networks

Matthias Rosenschon[1], Markus Heurung[1], Joachim Habermann (SM IEEE)[1]

[1]Fachhochschule Giessen-Friedberg, University of Applied Sciences,
Wilhelm-Leuschner-Strasse 13, 61169 Friedberg, Germany
matthias.rosenschon@iem.fh-friedberg.de

May 26, 2004

ABSTRACT

This paper addresses mobile ad-hoc network (MANET) simulations with the simulation tool network simulator NS-2. Since the simulation software does not include advanced protocols and sophisticated mobile network nodes new implementations have been performed. Firstly, a new mobile gateway has been introduced and secondly a standardised routing protocol has been enhanced to manage interaction between standard mobile nodes and the newly implemented gateway. Test simulations show that all modifications work properly and results are reasonable.

KEYWORDS

Split-level programming, NS-2, wireless networks, MANET, ad-hoc routing, AODV

INTRODUCTION

The project concentrates on routing protocols in wireless networks. Therefore, simulations have to be performed to investigate modifications of existing implementations and new routing protocols for wireless ad-hoc networks. The primary goal of our work is to provide ad-hoc networks access to the internet by a mobile internet gateway [9] which is both part of the ad-hoc network itself and part of the internet at the same time. To simulate mobile wireless ad-hoc networks an event driven open source software simulator called NS-2 [2] is used. In NS-2 various types of mobile nodes are implemented. However, a new mobile gateway node with two wireless interfaces as well as new protocol features, which support the access to the internet for any mobile node within the ad-hoc network, had to be implemented.

NS-2 uses an uncommon programming technique called split-level programming [4]. In NS-2 C++ is used to provide fast computable parts of network nodes that are glued together by the tcl scripting language. This style of programming combines both fast computation of compiled programs and flexibility of scripting languages in modifying node structures. If new protocols have to be implemented, both C++ objects and tcl scripts have to be modified. Since the documentation of NS-2 is in most cases incomplete, some additional work is typically required to understand the functionality principles of NS-2 in order to implement new agents, etc.

Figure 1 gives an overview over the network topology. A mobile node (MN) travels from one ad-hoc network (cluster of green dots) to another and holds connectivity through two gateways (GW) to a corresponding node (CN). Such a procedure is typically called handover. In contrast to other implementations, here the gateway is mobile and thus requires two wireless interfaces. All mobile nodes that are available in NS-2 up to now have only one wireless network interface. One interface of the gateway points to the inner ad-hoc network. The other interface is used for connectivity to the internet represented by the corresponding node.

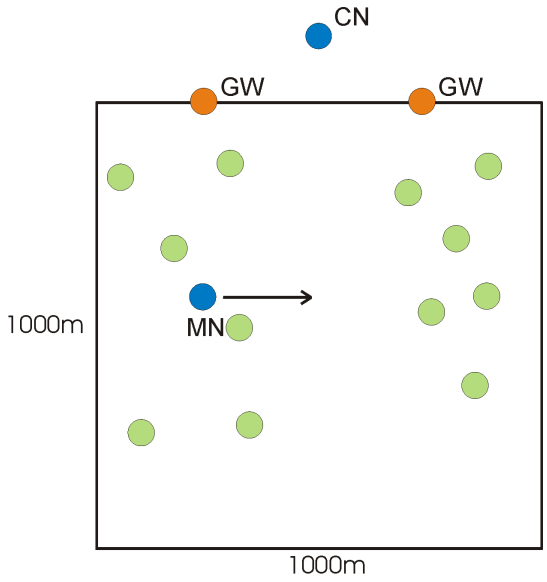

Figure 1. Mobile node moves between ad-hoc clusters

In NS-2 routing is provided by so called agents that are included in all nodes. Routing in an ad-hoc gateway requires a new agent, the AGWAgent, which had to be developed and inserted into the gateway node structure. The modifications of the node structure are discussed in chapter 1.

The Ad-hoc On demand Distance Vector routing protocol (AODV) [5] has been selected as ad-hoc routing protocol for all simulations. Several changes within the appropriate AODV routing agent were necessary to allow mobile nodes to find their corresponding gateway in an ad-hoc network [8]. Changes are discussed in chapter 2.

To test and approve all implemented code modifications of NS-2 a specific test scenario has been set up and simulated successfully. See chapter 3 for a verification test.

Chapter 4 gives a summary of the paper and an outlook to future work.

1 MODIFICATIONS ON NODE STRUCTURE

To discuss the necessity of modifications in NS-2 the simulation environment of Figure 1 is exemplarily used for all further considerations.

In NS-2 conventional mobile nodes consist of a number of C++ objects which are bond together by tcl-scripts (split-level-programming). In [2] the structure of a conventional mobile node is already available. The new mobile gateway, which has been obtained by extending the conventional mobile node is given in Figure 2.

Every box (square, ellipse, or trapezoid) is a C++ object. All objects are bond together via tcl-commands. Routing within the mobile gateway is carried out by the new object AGWAgent. It forwards data packets to the inner or outer wireless interface respectively, here marked with `inTarget_` and `outTarget_`. Two modified routing agents (`RTAgent`) then process all data and routing packets, in order to perform reasonable ad-hoc behaviour. For example, if no route to the target node exists, either a new route has to be found or a route through a gateway has to be discovered according to the modified AODV protocol [8].

Packets then move through the interface stack, reach the channel and will be delivered to the next node. There they pass the stack and are transported to the entry point (`entry_`). Then they pass address and port demultiplexers (`(addr|port) demux`) and reach the new AGWAgent of the next node or even finally the sink (`Src/Sink`), if packets arrive at the corresponding node.

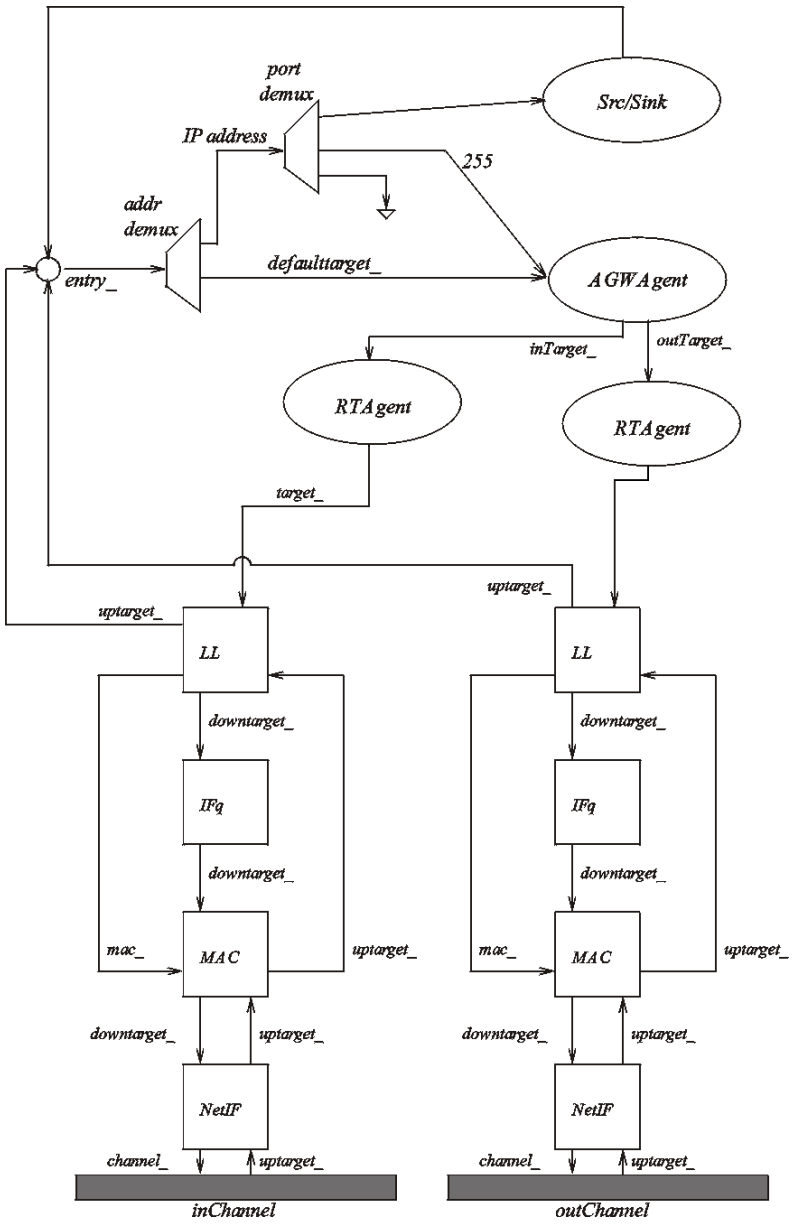

Figure 2. Structure of newly implemented mobile gateway

2 MODIFICATIONS OF AODV

Some major changes in the AODV routing protocol were necessary to obtain a behaviour that allows mobile nodes to find their gateway in order to get a valid connection to a corresponding node outside the local ad-hoc group. In principle this can be done in three different ways. One approach is that the gateway periodically sends advertisements to announce its presence, and the participants of the ad-hoc group (cluster) build their IPv6 address from that information [6]. The other way to obtain a routable address is that mobile nodes ask for a gateway by sending solicitations into the ad-hoc network that will be answered by the gateway. The third approach is a mixture of the first and the second. See [8] and [1] for gateway discovery details and [7] for address autoconfiguration.

As an example for modifications of the AODV routing agent two extended functions are discussed in the following: `recvAODV()` and `recvAdv()`. All discussed functions are declared and implemented in aodv.(h|cc) in the NS-2 source tree.

The unmodified function `recv()` receives all incoming packets (AODV routing and data traffic) and decides where to forward each packet. As a result all AODV packets are forwarded to `recvAODV()` that again decides by a switch-case statement what type of AODV packet has been received, standard AODV packets [5] or advertisement packets [8].

```
recvAODV ( packet ){
switch ( AODV type )
   AODVTYPE_RREQ:  recvRequest( packet )
   AODVTYPE_RREP:  recvReply( packet )
   AODVTYPE_RERR:  recvError( packet )
   AODVTYPE_HELLO: recvHello( packet )
   AODVTYPE_ADV:   recvAdv( packet )
}
```

Here a pseudo code language to shorten complicated C++ code is used in order to gain a clearer overview over the program code. The switch-case statement was extended by a new case for AODV gateway advertisement packets. The advertisement packets will be handled by the function `recvAdv()`:

```
recvAdv ( packet ){
   if ( i am packet's source )
   drop packet

//add or update route to GW (packet's source)
   if ( no route before OR
        newer route OR
        shorter OR equal route )
   update route

//add or update default route
   if ( no default route before OR
        newer default route OR
        shorter default route OR
        different gateway)
   update default route

   if ( advertisement not processed before )
   forward packet
 }
```

This example describes how gateway advertisement packets are handled. If the packet was originated by that node itself (compare Figure 2) it will be dropped. If that node has no route to the target node or the packet describes a newer or shorter route (or equal route which resets timeouts) to the target node, the appropriate route table entry will be updated. If the node has no default entry in its routing table (for a gateway) or the packet describes a newer, shorter, or equal route to a gateway, or a new gateway (e.g. after a handover), the route entry will be updated. If this advertisement packet was not processed before it will be forwarded to other nodes via the network interface (compare Figure 2).

3 SIMULATIONS

To test the correct functionality of the new implementations some test scenarios have been simulated. One of them is presented in Figure 3. It checks several functionalities of the newly implemented mobile gateway like gateway discovery with all three methods and the forwarding of packets by the AGWAgent.

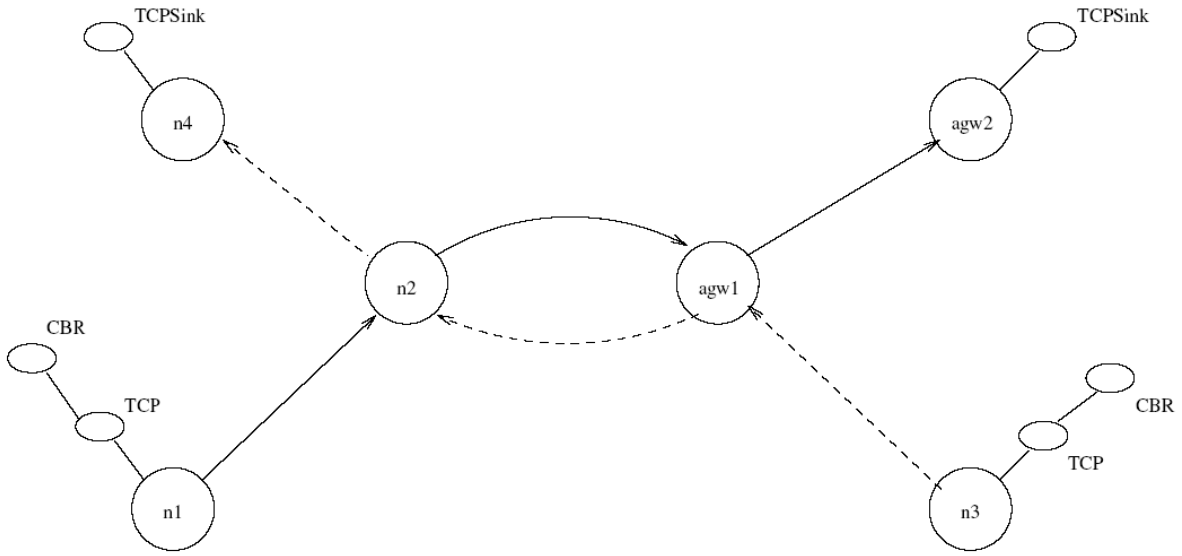

Figure 3. Test scenario for newly implemented mobile gateway

In the test scenario of Figure 3 four conventional mobile nodes and two ad-hoc mobile gateways are used. From the CBR (Constant Bit Rate) traffic source at node #1 (`n1`) a data connection via node #2 (`n2`) to gateway agw #1 (`agw1`) is established. This gateway forwards the data packets by its outer interface to gateway #2 (`agw2`) where the traffic sink is located. In reverse order a traffic connection between the CBR data source at node #3 (`n3`) via gateway #1 and node #2 to destination node #4 (`n4`) is set up to show that both directions are possible and working correctly. This has been asserted by examining the output trace files of that specific simulation.

More realistic scenarios have been defined to check if a handover of a moving mobile node between two mobile gateways is possible (see Figure 1). Again, output files show that all implementations are working correctly.

4 SUMMARY AND OUTLOOK

Due to the lack of proper documentation of the simulation software NS-2, modifications of existing protocols and implementation of new nodes may be quite tedious. Furthermore in NS-2 an uncommon programming technique, the split-level-programming, is used. Which combines the programming languages C++ and tcl.

In this paper implementation aspects of a new ad-hoc mobile gateway with two wireless network interfaces are discussed and modifications of the ad-hoc routing protocol AODV and its implementation into an appropriate agent within both a mobile node and mobile gateway are considered. Test scenarios have been performed to show that all implementations are working properly.

Future investigations will not only consider the mobility of the mobile nodes but also the

mobility of the mobile gateway and its corresponding mobile node cluster. Mobile gateways are then connected via a static basestation to the internet and provide access for all nodes within its cluster. Additionally, mobile nodes may change between different mobile gateways. Scenarios like these can be imagined when passengers get off a train, thus, leave a local and mobile ad-hoc group supported by a gateway within the train and perform a handover to a fixed access point mounted at the railway station without losing connectivity to their corresponding node and vice versa.

5 ACKNOLEDGEMENT

This work was supported by T-Systems under the German research project "IPonAir" [3], partly funded by the German Ministry for Education and Research (BMBF) and the Hessian Ministry of Research and Arts (HMWK).

REFERENCES

[1] Alex Ali Hamidian, *A Study of Internet Connectivity for Mobile Ad Hoc Networks in NS 2*, Department of Communication Systems, Lund Institute of Technology, Lund University, January 2003

[2] *The Network Simulator - ns-2*, http://www.isi.edu/nsnam/ns/, last visited may 26th, 2004

[3] *THE NEXT GENERATION WIRELESS INTERNET*, http://www.iponair.de, last visited may 26th, 2004

[4] John K. Ousterhout, *Scripting: Higher Level Programming for the 21st Century*, http://home.pacbell.net/ouster/scripting.html, published in IEEE computer magazine, March 1998, last visited may 26th, 2004

[5] Charles E. Perkins, *AD HOC NETWORKING*, Addison-Wesley, 2001, ISBN 0-201-30976-9

[6] Charles E. Perkins, et al., *IP address autoconfiguration for ad hoc networks*, Internet Draft, Nov. 2001. Work in progress.

[7] S. Thomson, B. Narten, *IPv6 Stateless Address Autoconfiguration*, Network Working Group, RFC 2462, 1998, http://www.ietf.org/rfc/rfc2462.txt, last visited may 26th, 2004

[8] Ryuji Wakikawa, et al., *Global Connectivity for IPv6 Mobile Ad Hoc Networks*, Mobile Ad Hoc Networking Working Group, http://www.join.uni-muenster.de/Dokumente/drafts/draft-wakikawa-manet-globalv6-02.txt, last visited may 26th, 2004

[9] B. Xu, S. Hischke, E. Weiss, *Integration of Ad hoc Networks with Wireless Access Networks using Ad hoc Gateways*, 9th Wireless World Research Forum Meeting, Zürich, Swiss, Juli 2003