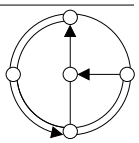
ETH
Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich



SS 2002 Roger Wattenhofer, Keno Albrecht, Fabian Kuhn, Aaron Zollinger

Mobile Computing
Exercise 6

Assigned: June 13, 2002
Due: June 27, 2002

1 Multihop neighborhood

In exercise 4 you implemented a broadcast PING/PONG scheme to discover your neighborhood (nodes that are within your radio range). Now we introduce two more message types to see what is behind the horizon.

The idea is to ask a known node about its neighbors. When you iterate this step you should be able to *crawl* the ad hoc net and consequently learn the ad hoc graph. It is mandatory to answer an incoming **GETNEIGHBORS** message by sending back a **MYNEIGHBORS** message, which includes a list of your current neighbors. Since you already know your neighborhood, it shouldn't be much work to put it into a byte array (remember that addresses are encoded little endian). The following tables summarize the new messages:

Message Type	Message Format (field size in bytes)
MYNEIGHBORS	type (1) — number of neighbors (1) — neighbor addresses (variable)

Message Type	Type Value	Reaction upon Receipt
GETNEIGHBORS	0x40	send MYNEIGHBORS to sender of GETNEIGHBORS
MYNEIGHBORS	0x41	whatever you want..., e.g. build graph

2 Route 66 – An awesome game

As promised you will finally implement *an awesome game* in this exercise. We will call it **Route 66** – just to emphasize that it will use the routing facilities that you already implemented in the last exercise.

We want to send messages from a server A to a server B. Since the servers don't necessarily *see* each other, we need intermediate nodes to forward the messages. Nothing in life is for free, and certainly intermediate nodes want to be paid for routing our messages. On the other hand we want to spend as little money as possible. So before sending the message, A will calculate a minimal cost route. Each node on this route will be paid the amount it asked for. To make it more exciting, server A and B are mobile, which makes it necessary for the intermediate nodes to move as well. Also the costs for routing a message don't need to be fixed – each node can make a different offer each time a message has to be routed. In this game you will play the role of an intermediate node. The goal of this game is to accumulate as much money as possible (however, as in real life we won't pay you at all ;-)). To do so, you need to find a good position between A and B and hope to have made the right offer.

How does it work in more detail? Server A (ID = 1000) will send messages in more or less regular intervals to B (ID = 1001). Before that it will broadcast a **NEXTTURN** message, that includes a timestamp (1 to 3 minutes in the future[1]) until which you have to make your offers for

[1] We will synchronize our clocks before the game starts.

routing the upcoming message. Some seconds (10 to 30) before the timestamp, A will broadcast a *Do Not Move* message (as a plain text SRMSG packet) as a reminder. After receiving this message just do so :-), don't move around anymore, but instead send a **MYOFFER** message to server A, which includes your offer (allowed integer values range from 1 to 66) for routing the next message, and additionally your current neighborhood (same as for MYNEIGHBORS). At the deadline (plus some *bonus* seconds for delay/latency) we use the neighborhood to calculate the mentioned minimal cost route, on which we will then send the message to B. If we succeed and the message arrives at B we will give each node on the route the amount it asked for.[2] To give you feedback on what's going on, we will broadcast a **SCORE** message, which contains a list of all participating users, their offers, current score, and the route taken. After a while we will send the next NEXTTURN and start over again.

The following tables summarizes the packets used for **Route 66.** Remember that all numbers are encoded little endian.

Message Type	Message Format (field size in bytes)
NEXTTURN	type (1) — timestamp (8)
MYOFFER	type (1) — offer (1) — number of neighbors (1) — neighbor addresses (variable)
SCORE	type (1) — number of players (1) — player ids (variable) — offers (variable) — score (variable) — route length (1) — route (variable)

Message Type	Type Value	Reaction upon Receipt
NEXTTURN	0x50	remember timestamp somehow
MYOFFER	0x51	..., you won't receive it (only addressed to server A)
SCORE	0x52	whatever you want, e.g. use to figure out strategy for next offers...

To keep it simple: To participate in a minimal way, all you have to do is to move around (or even stand still), react on the *Do Not Move* message and send a MYOFFER message. If you want to implement more, you might install a gui–timer for the NEXTTURN message, crawl your neighborhood by sending GETNEIGHBORHOOD messages and find out a good strategy based on the SCORE messages :-).

[2] If there are equally expansive routes we will make a random choice among those. If we don't succeed (because you moved or gave us a false neighborhood) we will try the next best route.