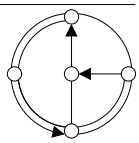
ETH
Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich



WS 2005-2006

Roger Wattenhofer, Nicolas Burri, Pascal von Rickenbach,
Yves Weber, and Andreas Wetzel

Mobile Computing
Exercise 5

Assigned: December 05, 2005
Due: December 19, 2005

1 Dynamic Source Routing Continued

In the last exercise we introduced a variant of Dynamic Source Routing (DSR). You have already implemented one major part of this algorithm, that is flooding. In this exercise you are going to complete the implementation of the algorithm.

As you remember, this is a completely demand-driven routing algorithm. Messages are only transmitted as necessary: There is no periodic message exchange. A source wishing to send a (user) message, first checks whether a route to the destination is already stored in its route cache. If so, the packet is equipped with the route and transmitted. The destination receiving the message will reply with an acknowledge message, reverting the route from the received message. If this message arrives back at the original sender, the latter can report successful transmission of the initial message. If however there is no cached route or if the message is not acknowledged within a certain timeout (for instance due to moving network nodes), the source will initiate route discovery. We suggest to first broadcast a *route request* message with TTL 1 in order to detect a neighboring destination without flooding the network. In case of failure, again after a certain timeout, retry by sending *route request* messages with TTL values 2, 4, and 8, as long as necessary. If still no route has been discovered, finally set TTL to 0, which will lead to flooding of the whole network. If again there is no positive answer, we give up and report an error message.

As in the last exercise we define different message types: *source route message* (SRMSG), *source route acknowledgement* (SRACK), *route request* (RREQ, listed for the sake of completeness), and *route reply* (RREP). The following table defines the packet format for these messages. For simplicity, we use the same format for every message type.

Message Type	Message Format (field size in bytes)
RREQ RREP SRMSG SRACK	type (1) | ID (2) | sender (2) | receiver (2) | ttl (1) | route index (1) | route length (1) | route (variable) | data length (2) | data (variable)

The type field contains the message type value (see below). The (message and flood) IDs are necessary to match corresponding messages and acknowledgements and to control flooding: Only a *route request* containing a (sender, flood ID) pair seen for the first time should be rebroadcasted or answered to with an RREQ message, respectively. The IDs are unsigned 16 bit numbers which should be generated increasingly and wrap around (start again from 0) when reaching `0xffff`.[1] The sender and receiver fields contain the addresses of the source and the destination of the complete route. The route index field holds a pointer to the next hop in the route. The route

[1] In order to cope with this wrap-around (and also relaunched applications—which therefore restart to increment their IDs at 0), a perfect implementation should contain some kind of "ID ageing", having "old" IDs lose their validity. In a preliminary version this problem can however be neglected.

length describes the length of the route. The complete route (including source and destination) consists of a sequence of 2 byte-addresses, starting with the source. Fields that are not used for a specific message type should be filled with zeroes.

The following table defines the message type values and summarizes the ways a routing node *must* react upon receipt of a message of according type. Again, in some cases additional local action will be useful, if not necessary.

Message Type	Type Value	Reaction upon Receipt
RREQ	0x11	if the destination is reached, reply with an RREP; otherwise: if TTL > 1, decrement TTL, append my address to the route, and rebroadcast; if TTL = 1, do not rebroadcast; if TTL = 0, rebroadcast leaving TTL unchanged (flood complete network)
RREP	0x12	if the destination is not reached yet, forward
SRMSG	0x21	if the destination is reached, reply with an SRACK; otherwise forward the message to the next node in the route
SRACK	0x22	if the destination is not reached yet, forward

A final hint: Choose the timeouts according to the maximum number of hops in a roundtrip (message → acknowledgement). Find a reasonable timeout for the "final" route discovery try (TTL = 0).