Trainspotting, a WSN-based train integrity system

Hans Scholten
University of Twente, Enschede, the Netherlands

Roel Westenberg, Manfred Schoemaker
Strukton, Hengelo, the Netherlands

Abstract

In contrast to classic train protection systems where most of the safety measures are built into the rail infrastructure, future versions of the European railway safety system ERTMS (European Railway Traffic Management System) require trains to check their own safety. One of the safety measures is the train integrity check, which checks whether all carriages are still in the train. The check has to be completed in a certain amount of time. This paper presents an overview for communication in such an integrity system, based on a wireless sensor network (WSN). Because a WSN with nodes in each carriage has a linear structure, existing communication protocols which assume a mesh-like topology do no suffice. Simulation shows that the proposed communication is feasible and has the necessary (soft) real-time properties to complete an integrity check in time. The protocol, based on the IEEE 802.15.4 network stack, is implemented and tested by Strukton.

I. Introduction

Many train safety systems exist around the world. In the Netherlands ATB and ATB-NG are used, Belgium and France use TBL/Crocodile/Memor and in Germany LZB is used. Although different on implementation level, most of them are based on the principle that trains cannot collide if they are not permitted to occupy the same section of track at the same time. So the railway lines are divided into blocks in which a train detection system detects the presence of a train. If a block is occupied by a train, a second train is not allowed to enter that block and will be stopped. These systems have many drawbacks. One of them is that systems are not compatible. Trains crossing borders need to be equipped with all applicable safety systems for countries they travel to. For example, the high speed Thalys has five different systems on board because it travels to France, Belgium, the Netherlands and Germany. Another drawback is the amount of rail side equipment for train detection, control systems and signals, which all has to be maintained and must be prevented from stealing (copper!). Yet another drawback is the poor utilization of the tracks, because block length is fixed. A more flexible system could adjust block length e.g. based on speed, weight and length of trains. To overcome these problems, a new European train protection system is being developed under the name ERTMS (European Railway Traffic Management System) [1]. There are three levels of ERTMS: the first level is similar to ATB-NG and doesn't offer many improvements except that it will be a European standard. It uses train detection and signaling on track, and (short range) communication with the locomotive is done via eurobalises. Eurobalises are antennas placed between de rails at regular intervals. ERTMS level 2 is more advanced: signal information is moved from the track to the locomotive, so trackside signals aren't necessary anymore. Signal information is exchanged wirelessly. The train reads its position from the eurobalises and the actual position is interpolated using speed sensors. This information is also transmitted to a control center where it can be used to calculate the necessary safe distance to the next train. Rail side train detection is still used in this level. In level three, the old fashioned train detection system using segments is removed from the track, and replaced by a moving block around the train which is dynamically determined: The train reports its location to the control center where the required safe distance to the next train is calculated. Because this level of ERTMS lacks rail side train detection, the whole system's safety depends on the location trains pass on to the control center. The required amount of trackside systems in ERTMS level 3 is reduced to a minimum. All parts of the train protection system are moved to the locomotives and a central system. However,

sometimes a carriage gets lost (disconnected) from the train. While in the old system the number of carriages is counted by the rail side detection system, in ERTMS level 3 the train itself must check its integrity at all times. It will report to the control center the moment it loses a carriage. In our approach all carriages are equipped with sensor nodes that measure movement. Integrity of the train is checked by measuring synchronicity of movement of the different carriages [2][3]. The topology of the train is calculated from the paths messages follow between locomotive and carriages. In the following we focus on the WSN communication, which differs significantly from existing WSN protocols because of the linear topology of trains.

II. Communication overview

Train communication features some rather specific characteristics for the protocol to be developed. Some are similar to general WSN [4], others are unique for the train integrity system.

Coordinator: The distinction between ordinary sensor nodes and special nodes is clear. The locomotive is the only special sensor node which initiates actions. Initially more than one 'coordinator' may be in the network, to one of them a carriage eventually will subscribe. *Routing:* Nodes only need to reach the locomotive and their direct neighbors. So routing algorithms can be kept simple; packets only need to be routed towards, or away from the locomotive. *Multi-path:* Packets will barely travel via different paths, because the sensor nodes will be situated in a more or less straight line. *Topology:* Only the locomotive needs to know the network's topology. *Synchronization:* Synchronization (normally considered an expensive procedure) can be combined with other communication, a message periodically sent from the last carriage for instance. *Sending/receiving:* Sending and receiving is regulated (nodes don't initiate communication themselves, besides the correlation data in the initialization phase. Other communication will only be in response to a received packet. This makes collision avoidance much easier. Complex collision avoidance techniques such as using time slots, allocated for each individual node, aren't necessary. *Traffic fluctuations:* The amount of traffic in the network does not depend on events from the environment, except for the initialization state. During operation, the traffic patterns are known on beforehand.

The train communication protocol is outlined in figure 1. The initialization is implemented in step 1 and 2: the sensor data correlation, and the topology discovery. By broadcasting sensor data at regular intervals, a sensor node will become aware of its neighbors presence. During the topology discovery, the request floods from the locomotive

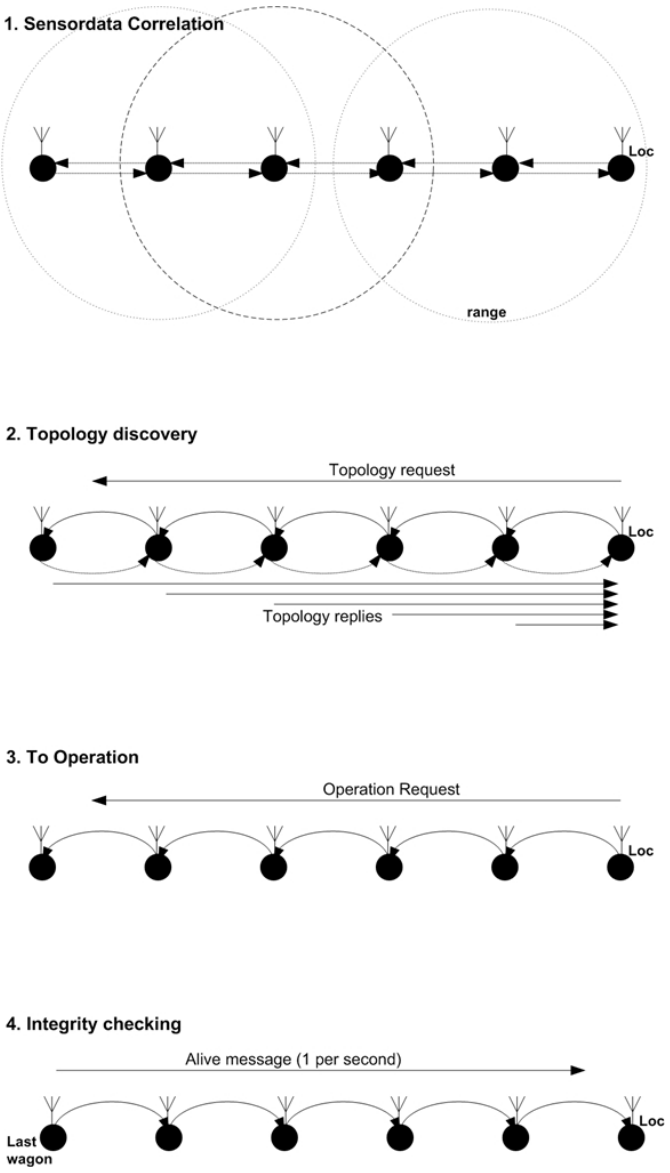

Fig. 1. Communication modes

into the network, in which individual hop counts are calculated. Each node replies to this message with a discovery reply, in which the node's neighbor information is contained. This way, nodes become aware of their neighbors, and the locomotive obtains an overview of their network, and the position of the carriages. At the request of the locomotive, sensor nodes can finally go into operation mode (figure 1-4), a state in which the train's integrity is continuously checked, and messages are routed from back to front at regular intervals. This is done with an operation request (figure 1-3), a message which contains all the carriages which will be part of the locomotive's network. One sensor node needs special mentioning, this

node, usually a node on the last carriage of the train, will be master during the operation state. This means that it will announce its presence at regular intervals (i.e. each second) with an alive message.

A. Range fluctuation and packet loss

Range fluctuation has a potential impact on both the communication's reliability and the validity of the estimated distances from sensor nodes to locomotives. Distance can be used to assess whether a sensor node will need to route a message because it is closer to the locomotive, or whether it should ignore the message because the sensor node is further away. So for the routing problem, the topology finding algorithm must take care of range fluctuations. The fact that a sensor node might be reachable once does not imply that this node will always be reachable and hence will be a reliable router. Depending on the average received signal strength, and the time for how a sensor node is visible, the protocol treats nodes differently. Figure 2 illustrates this. Figure 2a shows the ideal situation; every node has a fixed constant range, and only one neighbor can be reached. In reality however, due to range fluctuations, situations 2b and 2c may arise in bad or worst case scenarios. If during the establishment of the hop counts, node d is able to reach b, and not c (by extreme coincidence), this topology will be fixed, and node c will wrongly assume that it is further apart from d, then b (figure 2c). This will never lead to good routing results, so the protocol should be able to deal with these exceptional situations. The locomotive could for instance send these types of floods more than once into the network. Eventually the nodes will receive their correct hop counts. Once the topology is established, and nodes are aware of their position in the network, the protocol should contain an efficient procedure to deal with the fact that a message might not be received by any node considered to be able to route the message at all. The protocol must decide when messages are resent, and which nodes will participate in resending the message, without flooding the network.

B. Energy consumption

Sending and receiving messages is an expensive operation as far as energy consumption is concerned. So the time the radio is used should be limited, and during this time, resending packets should be prevented as much as possible. Still, reliable data delivery and low latency should be aimed at for this system. Sensor nodes usually provide low-power modes or stand-by modes, which should be taken advantage of as much as possible. In these low-power states, power consumption can be as low as a few microwatt. When the train is standing still, the system can

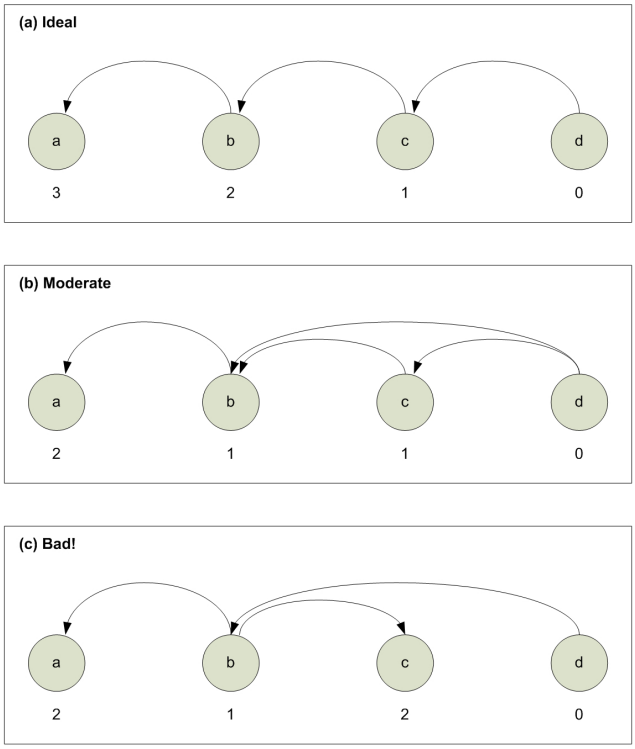

Fig. 2. Hop count calculation

go into a deep sleep mode. During initialization, information between neighboring carriages and locomotives will be exchanged continuously, so this state can hardly use any energy-conserving features. An effective approach to conserve energy is to minimize the time the radio is in the listening state. Very sophisticated MAC layer protocols which deal with this problem are for instance S-MAC [5] and LMAC [6]. These techniques synchronize all the sensor nodes and designate a collaborative sleep period. In between the sleep periods, communication is possible because all possible senders and receivers will be awake at the same time. This technique, though very sophisticated, isn't suitable for the train integrity network because in this specific situation, we don't want all the nodes to be awake at the same time, but we would rather provide an efficient linear path from the last carriage to the locomotive. For our application, when using the LMAC or S-MAC protocol, the problem arises that a node will not be able to receive a message, and forward it in the same short listen time interval. Instead. it inevitably waits for another sleep period before another sensor node can be contacted. Therefore, an approach similar to DMAC [7] is better suited. In DMAC, some node is considered the sink, towards many source nodes send messages. The awake intervals are ordered successively, where nodes closer to the sink wake up later,

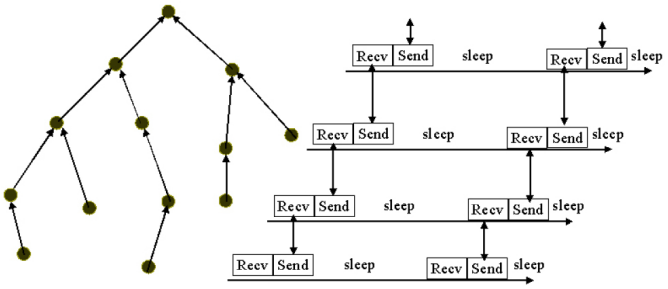

Fig. 3. DMAC data gathering tree

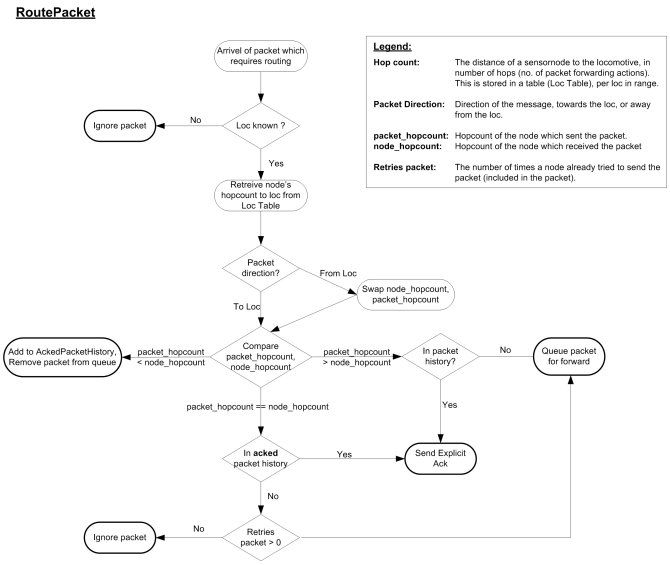

Fig. 4. Packet routing

thus providing an efficient path without sleep delays, as shown in figure 3.

C. Routing

Packets which require routing are processed using the routepacket procedure. Its flowchart is given in figure 4. Instead of an explicit acknowledgement when sending a packet, the protocol will, after having forwarded a message, receive the same packet again, this time sent by a neighbor closer to the destination. That way it knows that the message did not fail to propagate further through the network, and the node can stop retransmitting the packet. Under normal circumstances, the protocol assumes that packets can hop at least one hop closer towards the destination per retransmit of the packet. For the less ideal case where a node fails to reach one hop closer to the destination, nodes with the same hop count distance will also start forwarding this packet after a certain amount of

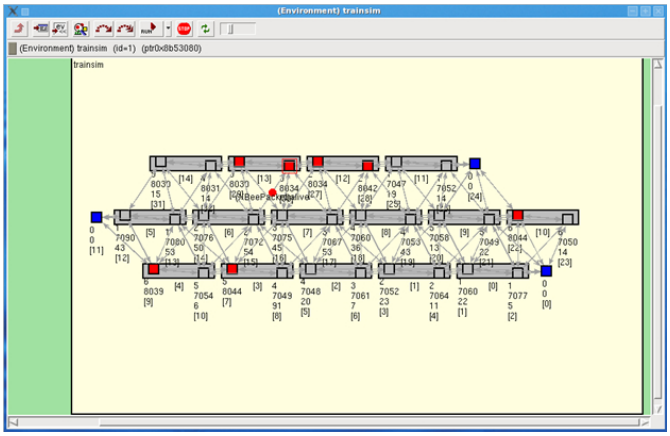

Fig. 5. Train topology in simulator

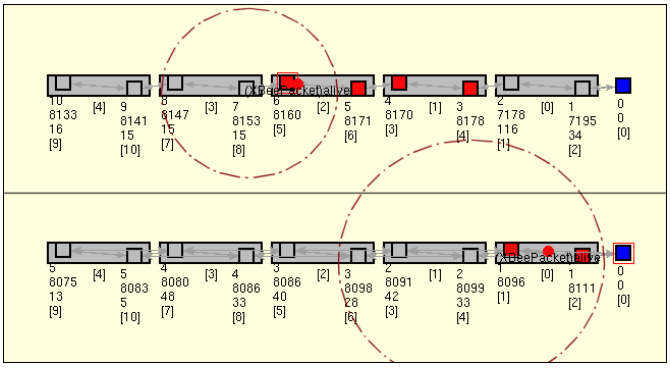

Fig. 6. Transmission range

retries. After all, it is very well possible that these nodes are (though at the same hop count level) in fact physically closer to the destination node. To prevent collisions caused by nodes at the same hop count distance trying to forward a message at the same time, the routepacket procedure can be extended with simple techniques such as a short random hold-off time, or a hold-off time dependent on the received signal strength. The latter can be used to optimize the packet forwarding, by letting nodes farther away (with a lower received signal strength) forward the packet quicker, as opposed to nodes closer to the sender.

III. Simulations

This section describes the Omnet++ simulation of the basic protocol discussed in the previous sections. Because Omnet++ is mainly used to simulate wired networks, it only supports direct communication amongst modules via channels. If we want to model wireless communication, we need to implement channels between one node and all the other sensor nodes in its range. This approach is for instance used by Dulman and Havinga in their Omnet++

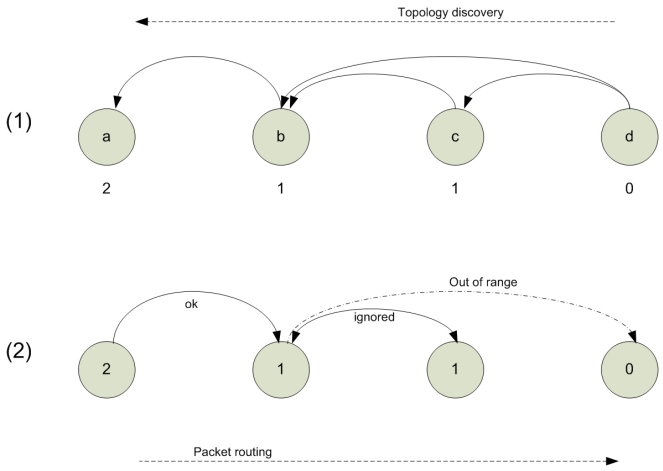

Fig. 7. Bad hop count discovery

simulation template for Wireless Sensor Networks [8]. The sending of one packet from the application layer, is modeled in Omnet++ by copying the packet and sending these copies over all individual links. Figure 5 shows the graphical representation of a random train topology. Each node has a position from which the connections with other sensor nodes are derived. Positions of nodes and interconnections are shown in this screen. During simulation, packets flowing through the network are animated. The rectangles represent carriages, equipped with two sensor nodes, and the single sensor nodes at the end of a row of carriages are the locomotives. Several parameters can be set on initialization of the simulation. For instance, the number of trains (with for each train the number of carriages), the positions of the trains, the range of the sensor nodes and the amount of packet loss. Figure 6 shows varying the range of the sensor nodes on one train. As expected, this changes the number of interconnections, and it affects the hop count per node. The hop count is illustrated in the simulation as the upper number in the list of four variables printed below each sensor node. The nodes address is the number below, between square brackets. The moment, and the interval at which the locomotives starts sending topology discovery requests into the network can also be regulated by setting parameters. It is very well possible that one flood will not reach every single sensor node in the network, or nodes may only be reached via detours, resulting in bad hop counts. Multiple floods solve this problem, so the duration of the initialization state, and the interval at which floods are initiated can be chosen freely as well.

A. Problems

Running the protocol in the simulator revealed some major shortcomings in the initial version of the protocol. Situations like illustrated in figure 7, where nodes receive messages from nodes with the same hop count, may arise after an initial discovery flooding in case of packet loss or extreme transmission ranges. This problem is solved by detecting if a node at the same distance continuously tries to forward a packet without result. In this case, nodes on the same level must eventually start forwarding the packet as well. Determining the last carriage, by sorting the modules on the train by hop count does not suffice in case of larger ranges for the sensor nodes. This may result in more than one node having the highest hop count. Right now, amongst these nodes, the node which is the farthest away is determined using average RSSI (signal strength), which might not always be the best solution. It solves the problem, as long as fluctuations in this parameter stay within certain limits.

B. Results

Simulations are carried out for a train composed out of 20 carriages (40 sensor nodes), for the duration of 30 minutes using an alive packet interval of 1 second. The situations in which one neighbor, two neighbors or three neighbors are reachable are simulated. The effect of this, in combination with different packet loss percentages are examined and the results are shown in figure 8. For different individual packet loss percentages, both the effect on the source-to-sink packet delivery, and the source-to-sink packet travel time are shown in the figure. In this simulation, the topology discovery is carried out without packet loss, in order to start the simulation with the correct source-node at the last carriage. Given ERTMS constraints, such as a deadline of four seconds for the alarm message, and a SIL (Safety Integrity Level) requirement of 99,90 we are able to judge whether the protocol performs sufficiently. For instance; given a very poor interconnection configuration of only one neighboring sensor node being reachable, we can derive from 8a that the individual packet loss percentage should be kept well below 25 %. If we assume a more realistic configuration, in which three neighboring sensor nodes are reachable, the individual chance for packet loss may be as high as 60 to 70 %! Next should be determined how long it takes the alive packets to travel from the back of the train to the locomotive. Figure 8b gives the average travel times for this, along with the standard deviation for this average. Note that after the last point on the curve, the travel time actually goes to infinity, because no packet is able to reach the destination anymore.

IV. Hardware implementation

The application's behavior in Omnet++ is programmed without any special Omnet++ or C++ features. To check the behavior of the developed protocol in practice, the same code is ported to a Rabbit [9] platform, equipped with MaxStreams XBee modules [10]. These radio modules were not chosen for their ZigBee functionality, which they can offer as well, but for the IEEE802.15.4 [11] firmware, with which they are delivered normally. This firmware offers the exact amount of functionality needed for our purpose; the physical and MAC layers, without any upper stack layers. During testing energy consumption is measured. Though the Rabbit platform is not designed for minimal power consumption, they performed rather well. On average, they are awake for a very short time, consuming 50mA@3.3V for the XBee module, and 25mA@3.3V for the Rabbit module. Asleep, the whole module consumes slightly less than 1mA@3.3V . Given that real sensor node platforms feature much better performance for energy consumption in both awake and standby states, this offers possibilities for this protocol. Using this platform, the roundtrip time of a message is 14 ms on average, and around 10 % of the packets gets lost.

V. Conclusion

In this paper we focused on the communication of a wireless sensor network based train integrity system for ERMTS. During different modes of communication the locomotive can determine the composition and integrity of the train. The current approach assumes that a carriage disconnected from the train by accident quickly disappears out of range, which results in absence of alive packets. Given the range of sensor nodes this is an appropriate solution. The current implementation of the protocol by Strukton has proven to work in the real world on a Rabbit3400-Xbee combination.

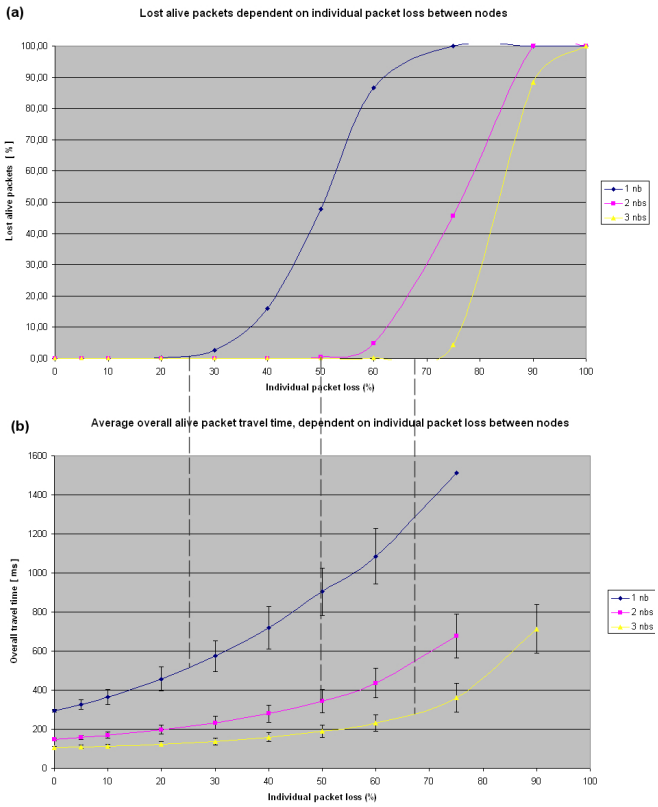

Fig. 8. Packet loss and overall travel time

References

[1] Ertms (European rail traffic management system): http://www.ertms.com.

[2] Mihai Marin-Perianu, Raluca Marin-Perianu, Paul Havinga, Hans Scholten: "Online movement correlation of wireless sensor nodes", 2007

[3] R.S. Marin-Perianu, M. Marin-Perianu, P.J.M. Havinga, J. Scholten: "Movement based group awareness with wireless sensor networks", 2007.

[4] Mario Strasser, Andreas Meier, Koen Langendoen, Philipp Blum: "Dwarf: Delay-aware robust forwarding for energy-constrained wireless sensor networks", 2007.

[5] Wei Ye; John Heidemann; Deborah Estrin: "An energy-efficient mac protocol for wireless sensor networks", 2002.

[6] L.F.W. van Hoesel and P.J.M. Havinga: "A lightweight medium access protocol (LMAC) for wireless sensor networks", 2004.

[7] Gang Lu, Bhaskar Krishnamachari, Cauligi S. Raghavendra: "An adaptive energyefficient and low-latency mac for data gathering in sensor networks", April 2004.

[8] S. O. Dulman and P. J. M. Havinga: "A simulation template for wireless sensor networks", Technical Report TR-CTIT-03-15, Enschede, April 2003.

[9] Rabbit semiconductor: "rcm3400 rabbitcore module", http://www.rabbitsemiconductor.com/products/rcm3400/docs.shtml.

[10] Maxstream: "xbee 802.15.4 rf module: http://www.maxstream.net/products/xbee/xbee-oem-rf-module-zigbee.php.

[11] IEEE 802.15.4 standard: http://www.ieee802.org/15/pub/TG4a.html.