Efficient Sensor Data Gathering and Resilient Communication for Rescue Scenarios

Daniele Munaretto[1], Chunlei An[2], Joerg Widmer[1], and Andreas Timm-Giel[2]

[1] DOCOMO Euro-Labs, Munich, Germany, `lastname@docomolab-euro.com`
[2] University of Bremen, TZI ikom/ComNets, Germany,
`{atg,chunlei}@comnets.uni-bremen.de`

Abstract. Wireless Sensor Networks (WSNs) have been used mainly to collect environmental data and send it to a base station. Routing protocols are needed to efficiently direct the information flows to the base station. Since sensor nodes have strict energy constraints, data gathering and communication schemes for WSNs need to be designed for an efficient utilization of the available resources. An emergency management scenario is investigated, where a sensor network is deployed as virtual lifeline when entering a building. In addition to navigation support, the virtual lifeline is also used for two purposes. Firstly, to exchange short voice messages between fire fighter and command post. For the communication between command post and fire fighter a fast and reliable routing protocol (EMRO) has been developed based on a broadcasting scheme. Secondly, for data gathering a network coding based algorithm has been designed. The feasibility of simultaneously using this virtual lifeline for data gathering and communications is investigated in this paper by means of simulation and real experiments. The resilience to packet loss and node failure, as well as the transmission delay are investigated by means of short voice messages for the communication part and temperature readings for data gathering.

1 Introduction

Sensor networks play an increasingly important role in emergency and rescue scenarios. For example, fire fighters today take a physical lifeline (rope) with them into a burning building, to be able to find the way back in poor visibility (smoke). However, it may happen that fire fighters on their way back run into fire and risk their life. Motivated by casualties in France, it was investigated in the wearIT@work project if an electronic, virtual lifeline consisting of sensor nodes can replace the physical lifeline. The virtual line has several advantages. First, it helps a fire fighter to orient himself without the hassle of being attached to a physical rope. Second, it allows to monitor the environment along the lifeline on the return path, and third it can be used for limited communication between the fire fighter and the group leader at the command post outside the building.

For indoor navigation and in particular to guide the fire fighter back out of the building, Pedestrian Dead Reckoning (PDR) is proposed as core technology. Based on accelerometers, the track into the building and thus a relative position can be estimated. However, the accuracy of the heading information is limited and the distance estimation suffers from irregular movement (e.g., crawling) of fire fighters in zero visibility environments. Over the distance the error accumulates significantly. Therefore hybrid technology is investigated including PDR and distance measurements to sensor nodes (RSSI-based or ultra-sound) of the lifeline [1, 2]. For real deployment of nodes a node dispenser is being developed in the wearIT@work project that can be mounted, e.g., at the back of a fire fighter oxygen bottle (see Fig. 1).

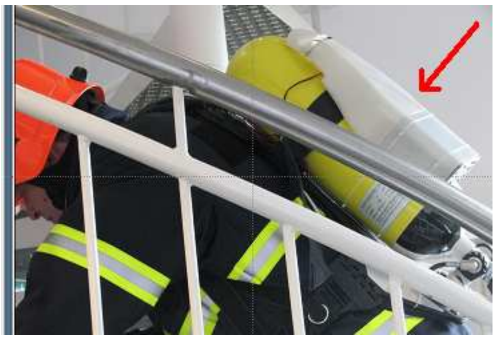

Fig. 1. Prototype of sensor dispenser for automated hands-free node deployment. Photo: BIBA,Germany

From a communication technology point of view, fire fighting is a very challenging application scenario. Fire fighters enter buildings with unknown communication infrastructure. The availability of cellular (e.g. UMTS, GSM) or wireless networks cannot be relied upon. Radio communication inside buildings with fire and smoke is possible, but degraded in vapor [3].

In this paper, the feasibility of using this virtual lifeline for communication is investigated through experiments and simulations. We focus on the performance of the sensor lifeline for sensing the environment and for the communication between command post and fire fighter. We specifically investigate the *joint* simultaneous use of the lifeline for direct communication and for the distribution of sensor data. For the distribution of sensed data, a network coding approach is implemented, which provides high loss resilience at a comparatively low overhead. For the data communication, a broadcast based scheme is developed, which is robust and yet efficient in topologies given by linear deployment of sensor nodes. Both approaches are described in detail in Sections 2 and 3. Finally, we want to clarify that communication through a virtual lifeline is designed just as a supplement to normal communication methods, e.g. GSM or WLAN.

2 Routing

Multi-hop routing protocols for ad hoc sensor networks [4] have been extensively studied in the past few years due to the difficulty to provide features such as self-organization and robust multi-hop routing [5]. Using a multi-hop routing protocol in WSNs, it is possible to gather data over a wide area with only one sink and to arbitrarily modify that monitored area by moving/adding/removing stations whenever needed. Since the stations are always monitoring their network neighborhood, these changes are quickly and automatically taken into account without the need to reconfigure the network. A station may also fail, for instance due to lack of energy, without impacting the data gathering. If that station was a part of a route to the sink, a new route will automatically be created and used to replace the deprecated one. The multi-hop routing schemes proposed for WSNs are to a great extend based on ad-hoc routing schemes as discussed in the IETF working group MANET [6].

In the specific case of a lifeline, sensor nodes are dropped sequentially by the sensor dispenser as the fire fighter moves. This sequence can be utilized for the routing protocol and allows for a much simpler and more robust protocol design. Node IDs are assigned when a node is being placed and as long as the fire fighter moves further into the building, nodes with lower node ID are closer to the command post. We propose a simple protocol EMRO (EMergency ROuting) based on broadcasting information in two directions: if a packet is sent by the fire fighter, it has to be forwarded to lower node IDs (towards the command post); if a packet is sent by the command post, it has to be forwarded to higher node IDs (towards the fire fighter). Each packet has a unique sequence number. Every time an intermediate node receives a packet, it will first compare this packet sequence number with the one previously stored in its memory. If the incoming packet has a higher sequence number, it will be forwarded and the intermediate node will update its status with the new sequence number. If a packet with lower sequence number is received, the packet is discarded. When the fire fighter sends data to the command post, any node receiving a message will rebroadcast it, provided that the message is received from a node with a higher node ID. In case the command post sends a packet to the fire fighter, intermediate nodes receive the message and wait for an ACK message from the fire fighter for a specified amount of time. If an intermediate node receives this ACK, it drops the current data message, since it has already reached its destination. If an intermediate node does not get any ACK before the timer expires, it retransmits the message. This helps to reduce redundant traffic inside the network. We also define several emergency messages for special alerts with high priority. If one of those messages is transmitted from the command post to a fire fighter, the intermediate nodes will forward it immediately. Because of the nature of broadcasting, the network is very robust to the addition/removal of nodes to/from the network. For a comparison, a conventional RSSI-based routing protocol is used. The protocol is a modified version of the routing algorithm used in the Sensor Scope project [7]. Each node sniffs the traffic of the neighborhood and in case the RSSI of the incoming packet from a neighbor exceeds a given threshold, that node is stored in a neighborhood list and a specific field called *cost to destination* is filled with 1 in case the fire fighter is reached in 1 hop, 2 in 2 hops and so on. This field is updated hop by hop by the visited nodes. The same applies to the command post, using a second field. For routing, nodes choose the neighbor with lowest *cost to destination* for routing packets to the fire fighter or command post.

3 Sensor Data Gathering

3.1 Network Coding

Network Coding (NC) [8] allows nodes to transmit packets that are combinations of multiple original packets, instead of simply forwarding the packets they receive or that originate at the node. For practical reasons, random linear network coding is often used. Coding operations involve addition and multiplication over a finite field. An outgoing packet Y_{out} is a random linear combination of the m coded or uncoded packets Y_{in}^{i} available at a node $Y_{out} = \sum_{i=1}^{m} k_i Y_{in}^{i}$. The packet can thus be written as a random linear combination of the original data packets $X^1,...,X^n$ to obtain $Y_{out} = \sum_{i=1}^{n} g_i X^i$. At the destination, decoding requires knowledge of the coding coefficients g_i, which can be transported, for example, in the packet header [9]. A node that has received a sufficient number of linear equations can decode by inverting the matrix of coding coefficients G to retrieve the original X.

3.2 Data Gathering in the Lifeline

Each node in a lifeline senses the surrounding environment and is responsible for disseminating these measurements, but may also forward other data such as voice messages. In this section we consider the problem of designing a coding algorithm which locally sets a suitable transmission schedule with respect to the sensing scope and communication range of the sensor nodes. Sensor readings are taken at regular time intervals and a node broadcasts them to its neighbors. When a node receives readings from its neighbors, it stores and combines the data received before broadcasting a new coded packet. Additional messages may be sent in case a dangerous situation is detected (e.g., high temperature, CO_2, smoke, etc.). Hence, the goal of the algorithm is to find the minimum number of transmissions required by the nodes to spread such readings throughout the lifeline so that each node can recover the sensed information. In particular, the fire fighter may pass by that node at a later time and may require these readings to determine if it is safe to proceed along the line. Furthermore, it may be useful for nodes to record the sensing history in their flash memory for longer term data gathering (with a low priority).

Two main issues need to be addressed: first, each node has a time varying transmit and receive range, due to changes in the transmit power and changes in the radio propagation environment; second, nodes may fail due to the limited battery life or to external causes (e.g., melting). The data gathering algorithm has to be designed to cope with these dynamics. We first address the issue of achieving a low energy consumption in the sensor network by maintaining a low number of transmissions (the largest source of power consumption). In [10], the authors consider the problem of finding the minimum number of transmissions needed in a ad-hoc network of N nodes for all-to-all broadcasting. For a range of different settings, they calculate the node's optimum number of transmissions C. However, this study is limited to the case of homogeneous settings where each node communicates only with its 1-hop neighbors. For a realistic network, it is necessary to extend the analysis to the case of heterogeneous node densities and radio ranges. We first consider a line topology where nodes are able to communicate with the $1,2,..,i$-hop neighbors, $i = 1,..,N$, in homogeneous settings. Then, we further extend the analysis to the heterogeneous case.

Homogeneous Settings According to [10], we model the network as an undirected graph $G = (V,E)$ with $|V| = N$ vertices, where the number of transmissions assigned to each node has to fulfill the cut conditions of the underlying graph as given in [10]:

$$\min \sum_{i\in V} C_i \tag{1}$$

$$\sum_{j\in N_S} C_j \geq |S|, \text{ with } 0 < |S| < N, \forall S \subset V \tag{2}$$

$$C_j \geq 0 \tag{3}$$

i.e., the number of transmissions over any cut of G, which partitions V into two sets S and $\overline{S}$, has to be at least as large as the number of nodes (and thus the number of information units) contained in the cut set[3]. N_S is the set of nodes that have an edge from S to $\overline{S}$. In case nodes communicate only with their 1-hop neighbors, the optimal number of transmissions C_i for node i in a line of N nodes is

$$C_i = \begin{cases} 1, & \text{for } i = 1, N \\ N - i + 1, & \text{for } i = 2, ..., (N-1)/2 \\ i, & \text{for } i = (N+1)/2, ..., N-1 \end{cases} \tag{4}$$

We adapt this analysis to settings where each node communicates with its r-hop neighbors, $r = 1,...,N$. Due to space constraints, we report only the case of an odd number of nodes; the analysis for an even number of nodes is a simple modification. We first consider the case $r = 2$, where each node may communicate with its 1- and 2-hop neighbors. Due to the doubled transmit range, each node increases the redundancy of information spread in its neighborhood by a factor of 2. As a first step towards the computation of the optimal transmission rates, we reduce the values in Eq. 4 by half, rounding up to the next integer. At the edge nodes, we leave the number of transmissions unchanged at 1, since they only have their own reading to spread in the line. Modeling the network as for the 1-hop case, the new transmission numbers have to fulfill the same cut conditions of the underlying graph as the original optimization problem given in [10].

Using half the transmissions of Eq. 4 and rounding up gives a feasible but not optimal solution. The optimum number of transmissions can be found through the following simple algorithm:

1. The edge node asks its 1-hop neighbor to reduce its transmissions by one. To fulfill the conditions in Eq. 2, the 2-hop neighbor increases its transmissions by one. The ripple effect propagates through the line, with each even neighbor reducing its

[3]For simplicity, we assume that a coding is only performed over a specific set of packets (called *generation*), and that a generation of packets is composed of one reading from each sensor at roughly the same time instant.

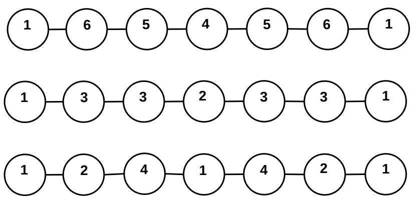

Fig. 2. Optimum transmission rates for 7 nodes on a line, starting from the 1-hop optimal values (top), reducing such values by half (middle), then after performing the iterations to find the optimum assignment (bottom).

number of transmissions by one and each odd neighbor increasing it by one. Since the edge nodes do not modify their transmissions, the number of modified even nodes is larger by one than the number of odd nodes, and thus the total number of transmissions is reduced by 1.

2. This process is repeated until the assignment of transmissions does not change any further.

Fig. 2 gives the steps to compute the optimal transmissions for an example topology with 7 nodes, where each node communicates with its 1- and 2-hop neighbors. The extension of the algorithm (including the initial division of the number of transmissions by the number of nodes covered by the transmit range in each direction) for settings with transmit ranges of $r = 3, \ldots, N$ is straight-forward.

Heterogeneous Settings We now consider the case of time-varying transmit ranges. Let Z_i be the set of nodes that is covered by a node i with transmit range r_i. Assume node j has a large transmit range. Clearly, if $Z_i \subseteq Z_j$ for any $i \in Z_j$, node j does not benefit from transmissions of i. In order to reduce the total number of transmissions in a lifeline, one should increase the number of transmissions of nodes with large transmit range and reduce it at nodes with smaller transmit range. The change of the number of transmissions is computed locally by each node based on a piggy-backed *feedback vector* sent with the data packets. Once a node j receives a packet from a neighbor i, it fills a vector at position i with value 1 in case the packet is non-innovative, 2 if the packet is innovative but cannot be decoded, and 3 if the packet is innovative and lets the node decode a new symbol. By default, the value of each entry is set to 0, which corresponds to the node not having received any packet from this neighbor. Once a transmission opportunity occurs, node j broadcasts a data packet, piggy-backing the feedback vector. Conversely, whenever node j receives a packet, it checks the value at position j of the incoming feedback vector. This provides the node with information about the previous packet it had sent. The information given by the feedback is used to assess the innovativeness of the node's own transmissions.

We now discuss how the algorithm works in practice. When a new generation of data (i.e., sensor readings from the environment) starts, each node broadcasts its own reading as uncoded information packet. After this first round, each node broadcasts packets coded over its own reading and the packets received from other nodes. Since feedback information about packet transmissions is obtained only in the following round, we ensure that feedback for all packets is given by sending a further packet after completing the assigned number of transmissions as dictated by the schedule. If the last packet sent is acknowledged as innovative, the node increases the transmission rate by one packet per generation, for the next generation of packets. The transmissions are reduced by one whenever the number of non-innovative packets sent is larger than one. We keep a safety margin of one additional packet to ensure that there is feedback for all required packets and as a precaution in case the topology changes or nodes fail. Reducing or increasing the transmissions only by one unit per generation caters to the time-varying nature of the transmit range. A gradual adaptation of the number of transmissions avoids that generations may frequently not be fully decodable when the radio environment is very dynamic.

Each generation involves the sensor readings of all the nodes generated within a certain time window. Thus, a new generation is started at the nodes at approximately the same time, which causes the nodes to broadcast their first uncoded packet. (This only requires a very loose synchronization among nodes.) To avoid collisions, packet transmit times are delayed by a small random offset. The following transmissions will occur periodically every $\frac{G_{time}}{C_i-1}$, where G_{time} is the expected recovery time of a generation. This leads to an even spacing of the C_i transmissions of a node. G_{time} can either be fixed before the deployment of the network or can be dynamically adapted by the nodes generation by generation. With this algorithm, some packets sent with a high transmission rate are not acknowledged by neighbors with fewer transmissions. In this case, nodes with lower transmission rate send feedback regarding the most recent packet received by their neighbor.

The feedback vector is also important for checking the connectivity of the network. When a node does not receive any packet from neighbors for a whole generation, it increases the transmit range by a fixed amount (see [11]). The same applies when the feedback vectors from neighbors have all entries set to 0, which means that the node can receive but it cannot reach the neighbors.

In summary, each node locally learns how to gradually adapt its transmission rate and transmit range. The scheme is decentralized, takes into account the time-varying nature of the node's transmit and receive range, ensures connectivity between remaining nodes in case of failures, and increases the node's life time by avoiding the transmission of non-innovative packets.

4 Simulations and Experimental Results

In this section we discuss our experimental and simulation results regarding the integration of the network coding based data gathering (Sec. 3) module with a broadcasting based routing protocol (Sec. 2). Experiments are performed on a lifeline composed of seven Tmote Sky sensor nodes [12]. They feature the Chipcon CC2420 radio chip [11] for radio communication (2.4 GHz, 250 Kbps). The radio chip is controlled by the TI MSP430 microcontroller (8 MHz, 10K RAM, 48K Flash). TinyOS [13] is used as sensor operating system. Due to the limited number of nodes available, we run simulations in TOSSIM, the TinyOS simulator [14], to evaluate the performance of our integrated

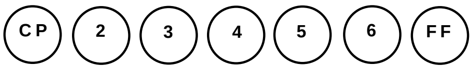

Fig. 3. Experiment Lifeline Setup

protocols for “X and Y” topologies (Fig. 5), where two lifelines cross each other (X) or merge at a point (Y). Concerning the data gathering protocol, we show how the algorithm dynamically adapts the nodes’ transmission rates and transmit ranges according to the actual network topology. For evaluating the broadcasting-based routing protocol, we compare it against a RSSI-based routing protocol. We analyze the robustness of the integrated protocols for this set of settings in terms of resilience to packet loss and node failure.

4.1 Line Scenario

We present the experimental results for a lifeline of 7 nodes, where 5 out of them are the sensing and forwarding nodes and the remaining two at the edges are the Fire Fighter (FF) at one side and the Command Post (CP) at the opposite side (Fig. 3). Each of the forwarding node runs the data gathering algorithm with a data transmission rate according to Sec. 3. FF and CP exchange short voice messages. When using a code rate of 20Kbps, a short voice message containing 7 words can be compressed to around 7 KB. The maximum size of a packet is 128 bytes and we use 110 bytes out of them for the data payload. Thus, we need 68 packets in total per voice message. The packets are transmitted at a rate of 5 packets per second. Setting a higher rate is detrimental to performance due to the higher number of collisions. The FF and CP use wearable computers (represented by laptops in the experiments) for displaying or playing out messages which are sent or received over the lifeline through a java interface.

The experiment evolves in 5 different steps. In step 1, all nodes communicate with their 1-hop neighbors and most of the time even with their 2-hop neighbors (we properly set the initial transmit ranges to ensure full connectivity between 1-hop neighbors w.h.p.). After letting the protocols run and reach a steady state, node 3 fails (step 2). At this stage we analyze the protocol’s resilience to node failure. At step 3, a further node (number 5) fails. Up to now, the lifeline is still able to maintain end-to-end connectivity for most of the time. But when node 2 fails at step 4, the protocol reacts by increasing the transmission power of the isolated nodes to reestablish connectivity. The same applies at step 5 when a further node (number 6) fails. At this last stage, only one forwarding node is alive and lets the FF and the CP communicate.

We define the transmission time as the time between the first packet being sent from the source and the last packet being received at the destination. The transmission rates are the rates for generating network coding packets (Sec. 3), which is simply background traffic from the point of view of the voice message transmissions. Fig. 4 shows how the algorithm adapts the number of transmissions that occur at each of the aforementioned steps. Note that we only show the final steady state of each step. As we can see from the graph, the algorithm well adapts the transmission rates after each failure. We compare the performance of EMRO against the RSSI-based multihop routing protocol, which is a modified version of the one used by the Sensor Scope project

[7]. Results in Fig. 4 show that the performance of EMRO is improved as far as delay is concerned. Regarding packet loss rate, the performances of these two protocols are comparable. We observe a slightly higher packet loss rate (overall packet loss rate inside the network) for EMRO, which is due to the redundancy inherent in the broadcasting. It is also related to the low buffer size in TinyOS 1. With the RSSI-based multi-hop routing, the performance in terms of transmission time and packet loss rate improves when more nodes fail, since the contention on the medium due to the multi-hop routing is reduced. Furthermore, the background traffic from network coding is lower as fewer nodes contribute to the sensor measurements.

4.2 X and Y Scenarios

The packet transmission rates used in the experiments must be changed in the simulator according to the new bit rates of 10 Kbps for the start symbols and of 40 Kbps for the payload featured by TOSSIM. Hence, sending a packet of, say, 128 bytes takes about 25.6 ms, which is substantially longer than the 4.1 ms required by real motes (without taking into account the channel busy time). As TOSSIM simulates a processor with a frequency of up to 4 MHz [15], which is much lower than that of a real TelosB mote, a node, in TOSSIM, is not capable to send packets as fast as in the real experimental environment.

Starting with the X setting, there are 2 FFs (node at the top-left as FF1 and node at the top-right as FF2) and accordingly 2 CPs (node at the bottom-right as CP1 and node at the bottom-left as CP2). In absence of intersections between these 2 lifelines, FF1 sends packets to CP1, and FF2 communicates with CP2 respectively. Each single

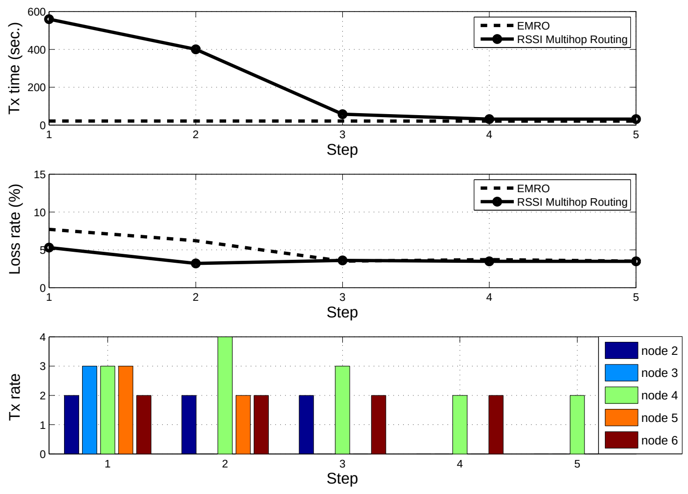

Fig. 4. Transmission rate adaptation, as well as loss rates and total transmission time for each step of the experiment. (a) and (b) for voice communication, (c) for data gathering.

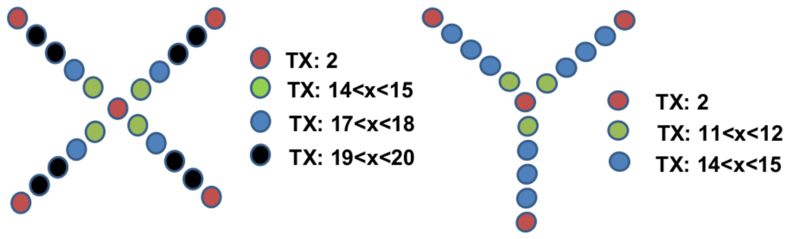

Fig. 5. Simulation results: transmission rate adaptation in X and Y topologies (1-hop communications): respectively, 2 crossing lifelines and 2 merging lifelines. At each node corresponds a color reporting the number of packets to be transmitted.

lifeline needs 34 seconds (Table 1) for transmitting a short voice message, which consists of 68 TinyOS packages, as aforementioned. In the setup shown in Fig. 5, the node in the middle is shared by both lifelines, therefore it should have double the amount of traffic. However, we decided to decrease the distance between the middle node and its 4 neighbors in the X topology and its 3 neighbors in the Y topology, so that the data load of the two lines can be shared by all these "core" nodes, avoiding loss of connectivity in case the middle node fails (which can be the result of the huge amount of data to be processed by only one node). Without such accommodation at the "core", the reaction time required to adapt the transmit range in order to recover from connectivity losses could be too long[4].

As a remark, the 2 FFs are scheduled to send packets in an interleaving manner to avoid collisions at the intersection node. With the aforementioned accommodations, each lifeline has an increased transmit duration of 68 seconds (Table 1), which is almost twice the time needed for single lifeline transmission.

Given the X setting in Fig. 5, both CPs are able to receive packets from both FFs. This kind of information redundancy can be utilized to increase the reliability of connections between FFs and CPs. Furthermore, due to the NC based data gathering spreading data along both the lines, the robustness of the X network is increased as well as data persistence. Even if one of the two crossing lines (X topology) fails, the other line can still combine (through coding) in its packets the information which came from the other line before it failed and decode it at the other CP.

Without loss of generality, the Y type scenario shown in Fig. 5 can be considered as an extension of the X type scenario. If one FF loses connection to a CP, it still has a chance to get contact to the same CP (or to another CP) by merging its lifeline into another one.

Simulation results are shown in Fig. 5 and Table 1. For each scenario the simulation is repeated several times, and the average values are reported. Absolute values of the transmission delay are not of interest, but they indicate the increment of time needed for transmitting a short voice message, when 2 FFs are trying to send messages at the

[4]In reality, sensor nodes should also be deployed in this manner, to ensure sufficient resilience against node failure.

	X Setting	Y Setting	Single Lifeline
Packet Sending Frequency (Hz)	1	1	2
Tx Delay (s)	68.137	68.465	34.765
Packet Loss Rate (%)	0	0	0

Table 1. Simulation results: short voice message transmission in X and Y settings and single lifeline.

same time. Furthermore, since the frequencies at which packets are sent out are chosen purposely to avoid channel and nodes' overloading, no packet erasures are experienced.

In Fig. 5 simulation results show that nodes' transmission rates fit well with the theory (even when increasing the transmit range to consider also 2- and 3-hop communication models, but results are not shown due to the space constraints). The nodes at the edges as well as the node in the middle, for both X and Y settings, just need to send the minimum amount of packets per data generation. The nodes around the middle node for both topologies share the load of the coded data coming from the lines so that, in case of node failure in the middle of the network ("core", i.e. middle node and its 1-hop neighbors), the connectivity is maintained.

5 Conclusions

In contrast to conventional WSN routing protocols, the proposed broadcasting based protocol has the advantage of adapting to changes in the network topology very quickly. This feature is beneficial in scenarios with mobile fire fighters and frequent node failures. In the X and Y topologies, 2 lifelines can coexist with each other well, and a fire fighter can immediately recover from the lack of connectivity with the command post by merging his lifeline with another one. The data gathering protocol running in the background has no negative impact on the transmission of short voice messages or alerts. Like the routing protocol, it is very well suited for the dynamics of our settings, and nevertheless operates at a comparatively low overhead.

References

1. Klann M, Riedel T, Gellersen H, Fischer C, Oppenheim M, Lukowicz P, Pirkl G, Kunze K, Beuster M, Beigl M, Visser O, Gerling M (Sep 2007) LifeNet: an Ad-hoc Sensor Network and Wearable System to Provide Firefighters with Navigation Support, *Ubicomp*.
2. Fischer C, Muthukrishnan K, Hazas M, Gellersen H (Sep 2008) Ultrasound-aided pedestrian dead reckoning for indoor navigation, *First ACM International Workshop on Mobile Entity Localization and Tracking in Gps-Less Environments*. San Francisco, California, USA.
3. Hofmann P, Kuladinithi K, Timm-Giel A, Goerg C, Bettstetter C, Capman F, Toulsaly C (Apr 2006) Are IEEE 802 Wireless Technologies Suited for Fire Fighters?, *12th European Wireless 2006*, Athens, Greece.
4. Al-Karaki J N, Kamal A E (2004) Routing Techniques in Wireless Sensor Networks: A Survey, *Wireless Communications, IEEE*.
5. Becker M, Schaust S, Wittmann E (July 2007) Performance of Routing Protocols for Real Wireless Sensor Networks, *International Symposium on Performance Evaluation of Computer and Telecommunication Systems (SPECTS 2007)*, San Diego, USA.
6. Mobile Ad-hoc Networks http://www.ietf.org/html.charters/manet-charter.html
7. SensorScope website http://sensorscope.epfl.ch/

8. Ahlswede R, Cai N, Li S Y R, Yeung RW (2000) Network information flow, *IEEE Transactions on Information Theory*, vol 46, pag. 1204–1216, num 4.
9. Chou P A, Wu T, Jain K (Oct 2003) Practical network coding, *41st Allerton Conf. Communication, Control and Computing*, Monticello, IL, US.
10. Loyola L, De Souza T, Widmer J, Fragouli C (Jan 2008) Network-Coded Broadcast: from Canonical Networks to Random Topologies, *NetCod 2008: Fourth Workshop on Network Coding, Theory, and Applications*, Hong Kong, China.
11. CC2420 data sheet http://www.ti.com/
12. Tmote Sky data sheet http://www.sentilla.com/moteiv-endoflife.html
13. TinyOS home page www.tinyos.net
14. Levis P, Lee N (June 2003) TOSSIM: A Simulator for TinyOS Networks, Version 1.0.
15. TinyOS lesson - using TOSSIM http://www.tinyos.net/nest/doc/tutorial/tossim-lesson.html