A Real-Time and Robust Routing Protocol for Building Fire Emergency Applications Using Wireless Sensor Networks

Yuanyuan Zeng
School of Power & Mechanical Engineering
School of Electronic Engineering
Wuhan University
Wuhan, China
zyywhu@gmail.com

Cormac J. Sreenan
Mobile & Internet Systems Laboratory
Department of Computer Science
University College Cork
Cork, Ireland
cjs@cs.ucc.ie

Lanny Sitanayah
Mobile & Internet Systems Laboratory
Department of Computer Science
University College Cork
Cork, Ireland
ls3@cs.ucc.ie

***Abstract*—Fire monitoring and evacuation for building environments is a novel application for the deployment of wireless sensor networks. In this context, real-time and robust routing is essential to ensure safe and timely building evacuation and the best application of fire fighting resources. Existing routing mechanisms for wireless sensor networks are not well suited for building emergencies, especially as they do not explicitly consider critical and rapidly changing network scenarios. In this paper, a novel real-time and robust routing protocol (RTRR) is presented for building fire emergency applications. It adapts to handle critical emergency scenarios and supports dynamic routing reconfiguration. Simulation results indicate that our protocol satisfies the criteria necessary to support building emergency scenarios.**

Keywords- wireless sensor networks; building fire; real-time; robustness; power adaptation

I. INTRODUCTION

In the near future buildings will be equipped with a range of wireless sensors as part of an overall building management system to detect and react to building fires [1]. In this context there is a need for real-time and robust message delivery, in the face of a network topology that can change rapidly due for example to node failure. However, most existing routing protocols consider energy efficiency and network lifetime as the foremost design factors. For example, related work on tracking forest fires does not consider evacuation of building occupants and guidance of fire personnel. This combination of real-time requirements coupled with changing network topology in a critical application scenario provides motivation for our research.

We propose a real-time and robust routing mechanism (RTRR) for building fire emergency using wireless sensor networks (WSNs). Here, robust means routing can be reconfigured quickly during the emergency, including route recovery and avoidance of routing holes due to node failures. Our approach avoids the need for location information or time synchronisation. To the best of our knowledge, this is the first time a real-time and robust routing mechanism for building fire emergency using WSNs has been proposed. We believe this protocol is useful in a range of WSN emergency applications.

The NEMBES project is funded by the Irish Higher Education Authority under the PRTLI-IV programme.

II. BACKGROUND AND RELATED WORK

Some WSN applications require real-time communication. For example, SPEED [2], MM-SPEED [3], RPAR [4] and RTLD [5] were designed for real-time applications with explicit delay requirements. Nevertheless, these routing protocols are not well suited for routing in emergency applications such as building fires, where critical and dynamic network scenarios are key factors. In this regard, the work by Wenning *et. al.* [6] is relevant - they propose a proactive routing method that is aware of the nodes' destruction threat and adapts the routes accordingly.

Other researchers work on emergency guidance and navigation algorithms with WSNs for buildings. Tseng *et. al.* [7] proposed a distributed 2D navigation algorithm to direct evacuees to an exit while helping them avoid hazardous areas. Based on this, Pan *et. al.* [8] proposed a novel 3D emergency service that aims to guide people to safe places when an emergency occurs. Barnes *et. al.* [9] presented a distributed algorithm to direct evacuees to an exit through arbitrarily complex building layouts in emergency situations. They find the safest paths for evacuees by taking into account predictions of the relative movements of hazards, i.e., fires and evacuees. Tabirca *et. al.* [11] solved a similar problem but under conditions where hazards can change dynamically over time.

There are many robust routing protocols proposed for WSNs. Deng *et. al.* [10] proposed a routing mechanism that can discover new routes after random failure nodes. The "Routing Hole Problem" is a very important and well-studied problem. Some existing "face routing" algorithms have been developed to bypass routing holes using geo-routing algorithms. Another way to avoid routing holes is to "jump" over the hole as proposed in [5].

III. DEFINITIONS

Given a homogeneous WSN deployed in a building with N sensors and M sinks. Each sensor can adjust its transmission range to one of the k levels: $r_0, r_1, \ldots, r_{k-1}=r_{max}$ using different transmission powers $p_0, p_1, \ldots, p_{k-1}=p_{max}$. Initially, all sensors use p_0 to minimise energy use.

T_{max} is the maximum acceptable delay of routing from node to sink, which is defined for the specific application scenario. Each sensor *i* will report data packets to a sink such that:

(1) a communication path from sensor to the sink can be found if such path exists,
(2) the end-to-end delay of the path is no more than T_{max},
(3) the choice of route is adaptively changed in response to failed nodes (assumed to be caused by fire), and
(4) a minimised power level ($min\{p_0, p_1, \ldots, p_{k-1}\}$) is chosen to ensure transmission satisfies (1), (2) and (3) without unnecessary power dissipation.

Each node in the network exists in one of four states:

(1) *safe*: node's initial state while no fire occurs,
(2) *lowsafe*: it is one-hop away from an "*infire*" node,
(3) *infire:* when it detects fire, or
(4) *unsafe*: it cannot work correctly due to a definite fire.

Each sensor changes its state autonomously according to fire impact and in response to received messages. A STATE message is used to propagate the current node state to its neighbours in a fire.

(1) STATE (INFIRE) message: If a sensor detects the fire, it enters "*infire*" by broadcasting the message to denote a new local fire source.
(2) STATE (LOWSAFE) message: "*Safe*" nodes that receive a STATE (INFIRE) message will become "*lowsafe*" and notify its neighbours. Other nodes that receive STATE (LOWSAFE) message only ignore it.
(3) STATE (UNSAFE) message: An "*infire*" node works until it cannot work correctly, then it becomes "*unsafe*". A node also becomes "*unsafe*" and broadcasts a STATE (UNSAFE) message if its energy is too low to work.

IV. RTRR Protocol Description

A. Initialised Routing Structure

1) Sink Beacon: We assume that sinks are deployed in a relatively safe place that cannot easily be destroyed. Each sink periodically generates a HEIGHT message using p_0. This serves to advertise to neighbour nodes and includes a "*height*" parameter that represents the hop count toward the sink and is initialised to 0. The height value is incremented by each forwarding hop. Each node records the height information in its local neighbourhood table when it receives the first HEIGHT message.

2) Delay Estimate: In the HEIGHT message propagation process, the sink-to-node delay (denoted as *delay* (*sink*, *i*)) is calculated by noting the cumulative delay on each hop. Our protocol does not assume any specific medium access control protocol, and so this delay value is simply an estimate. As messages propagate, the delay experienced on the current hop is calculated, updated locally and recorded in the HEIGHT message. The *delay* (*sink*, *i*) is also recorded in the neighbourhood table of each node.

We denote *T* (*i*, *sink*) as the estimated delay from a node to the sink. In WSNs, data is reported from nodes to the sink, while less traffic such as control command is delivered by the sink to nodes. However, since there is just a single transmission queue, it is not unreasonable to assume that the queuing delay is independent of whether a packet is going to or from the sink. We use *delay* (*sink*, *i*) to estimate delay from nodes to the sink in routing discovery to find a route that is likely to meet a lower delay threshold, i.e., using *delay* (*sink*, *i*) to estimate *T* (*i*, *sink*). If necessary, nodes can increase power and thereby reduce the realistic delay, while keeping the estimated delay as the base upper bound. In this way, we can provide a high probability of real-time delivery from nodes to the sink.

B. Routing Mechanism Details

1) Forwarding Choice: Each node in the neighbourhood table is associated with a *forward_flag* and a *timeout*. The flag is used to identify the best next-hop forwarding choice. The *timeout* is the valid time for the current forwarding node to prevent stale neighbourhood information. If the *timeout* of a forwarding choice expires, its forwarding flag is set to 0 to evict the stale relay node.

In order to select the best forwarding choice from the local neighbourhood table:

(1) we choose nodes with lower height,
(2) then we choose nodes with sufficient *slack* time, based on the estimated residual time to the sink,
(3) then we filter the remaining choices by node state in the priority from "*safe*" to "*infire*", and
(4) if there is more than one node that satisfies, we select the best forwarding choice with the highest residual energy. Finally, if there is still a tie, we choose the lowest node ID.

If we cannot find a suitable forwarding choice with the current transmission power, we say that a "hole" exists, i.e., we are stuck in a local minimum.

2) Hole Problem: The solution is to increase the transmission power gradually by levels to find another neighbour or to invoke a new neighbour discovery. If we can find a node in the neighbourhood table by adapting the transmission power, we increase the power level and name this neighbour as a forwarding choice. If this fails, a notification message is sent to its upstream node (i.e., its parent) to stop sending data packets to the current node. Then, a routing recovery is invoked. We increase power gradually but not directly to the maximum level because the larger the power, the larger the interference (and energy use). Moreover, it is common in today's sensor nodes to have only a handful of transmission power levels.

Fig. 1 illustrates two sinks and eight sensor nodes. The number on each node represents the "height" of the node toward the nearest sink. Node *i* reports data to *sink1*. As the route {*i*, *a*, *sink1*} with power level p_0 is invalid because the *slack* does not satisfy the estimated delay; so node *i* is in the "hole". If there are no existing eligible neighbours, *i* increases its power to p_1 to reach node *j* and delivers packets

to *sink2* using route {*i, j, sink2*} if the *slack* on this route is no less than the estimated delay.

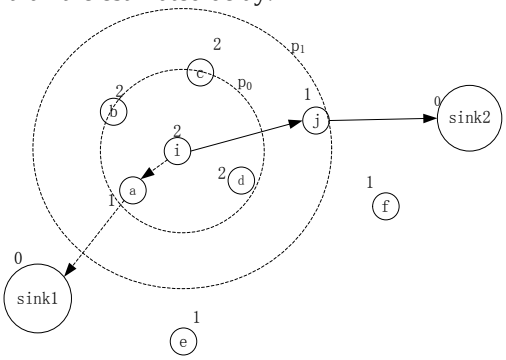

Figure 1. New neighbour discovery to solve routing "hole"

Recall that a sensor has *k* power levels $p_0, p_1, \ldots, p_{k-1}$ and can be in *k* levels of transmission range $r_0, r_1, \ldots, r_{k-1}$. We defined a formula to increase the transmission power:

$$p = p_{cur+\iota+1};\ \iota = 1, 2, 3, \ldots, k\text{-}1 \quad (1)$$

cur is the current transmission range level, ι is the count of unsuccessful attempts. A sensor will increase its transmission power gradually if it cannot find an eligible new neighbour. A node increases its power according to formula (1) until one of the following conditions is satisfied:

(1) it finds a node as a forwarding choice in "*safe*" state and that satisfies the height and estimated delay, or
(2) when $p = p_{max}$. In this case, it either finds a new neighbour as a forwarding choice using the height, estimated delay, and in a priority from "*safe*", "*lowsafe*" to "*infire*". Otherwise, no eligible new neighbour is found.

In the new neighbour discovery process, node *i* will broadcast a Routing Request (RTR) message in which it piggybacks *height, slack* and the newly adapted power p_1. If a node *j* receives the message and the estimated end-to-end delay is no more than *slack*, its height is lower than *height* (*i, sink*), and its state is "*safe*", *j* is selected as a new neighbour. If *j* receives the RTR with p_{max} and its height is lower than *height* (*i, sink*), *j* is selected as a new neighbour when *j* is not in the "*unsafe*" state. *j* will reply to *i* using the same power that *i* is using, after a random backoff to avoid collisions. The forwarding choices send a reply message using *power*(*i*) merely to reach *i*. For communicating with other neighbours they revert to their previous power level. Upon receiving the reply, *i* inserts the new neighbour into its neighbourhood table. During the RTR and reply message exchange, we can estimate the delay between *i* and its new neighbour *j* as follows:

$$Ave\_delay\ (i, j) = Round\_trip\_time\ /\ 2 \quad (2)$$

For meeting a real-time constraint, the forwarding choices should satisfy that *slack* is no less than the average delay between *i* and *j* plus the estimated delay at *j*:

$$Slack\ (i) \geqslant Ave\_delay\ (i, j) + delay\ (sink, j) \quad (3)$$

If there is more than one new neighbour found, the best forwarding node is selected using the priority of the state from "*safe*", "*lowsafe*" to "*infire*". If there is still a tie, the best relay is selected as the node with the highest residual energy and the lowest node ID.

A node decreases the transmission power to improve energy efficiency and network capacity when the delay deadline is well satisfied. So, when a node detects good connectivity with a safe node that is larger than a predefined threshold, i.e., $|Neighbour_{safe}| > N\_threshold$, the power decrease process is invoked.

We defined a formula to decrease the power as follows:

$$p = p_{cur-\iota'};\ \iota = 1, 2, 3, \ldots, k\text{-}1 \quad (4)$$

cur is the current power level and ι' is the count of decrement. A node is eligible for power decrement until:

(1) the minimum power has been reached,
(2) there are two consecutive power levels such that at the lower level the required delay is not met but at the higher level the required delay is met, and
(3) there are two consecutive power levels such that at the lower level the required safe neighbourhood connectivity *N_threshold* is not met but at the higher level it is met.

C. Routing Recovery

We assume that: (1) the minimal time interval between "*infire*" and "*unsafe*" state of a node is a parameter known as t_{unsafe}, and (2) we use necessary transmission range for connectivity between nodes (according to the selected power level) to approximate the minimum fire spreading time between two nodes. When a forwarding choice is used for routing, we add a *timeout* to avoid the use of stale and unsafe paths, i.e., every node on the path from source *s* to destination *d* has a *timeout* to record the valid time of each link on this route. The *timeout* is updated when node state changes occur among the neighbourhood. The forwarding choice that exceeds the *timeout* is considered invalid and then evicted. We assign an initialised large constant value to represent the estimated valid time for the node in "*safe*" state.

When a neighbour node *j* is caught in fire, a STATE (IN-FIRE) message is broadcast. If a "*safe*" node *i* receives the message from its neighbour, then *i* enters the "*lowsafe*" state. The *timeout* of *i* is updated as the minimum time this node may be caught in fire until it is cannot function:

$$timeout\ (i) = min\_spread\_time\ (i, j) + t_{unsafe} \quad (5)$$

The *timeout* of both downstream and upstream links that are adjacent to *i* are also updated accordingly. If *i* becomes "*infire*", the *timeout* of adjacent links are updated as t_{unsafe}, i.e., $timeout\ (i) = t_{unsafe}$. However, if *i* becomes "*unsafe*" by local sensed data and threshold, *timeout* (*i*) and the *timeout* of the adjacent links are set to 0.

The link *timeout* is updated as the state of the node adjacent to the link changes. When a node state changes in the fire, the upstream and the downstream links that are adjacent to this node will update the *timeout* on both links. For each link (*i, j*), the *timeout* for this link is calculated as:

$$timeout\ (path\ (i, j)) = \min(timeout\ (i), timeout\ (j)) \quad (6)$$

timeout (*i*) and *timeout* (*j*) represent the valid time for *i and j* in fire.

In a building fire, node failures due to fire damage will trigger routing tree reconfigurations. If a link *timeout* is lower than a threshold (i.e., the route will be invalid soon), a route reconfiguration is invoked to find another available path before the current one becomes invalid. The reconfiguration is only invoked by an upstream node *i* of the link (*i, j*) whose valid time is no less than the *timeout* of the link, i.e., *timeout* (*i*) ≥ *timeout* (*path* (*i, j*)). The routing reconfiguration of the node is invoked as a routing recovery by broadcasting a RTR message to set up a new route search. The search of the forwarding choice is invoked in its neighbourhood table to find whether one of the existing neighbours is eligible to act as a relay or not by adapting the power to the setting recorded in local neighbourhood. Otherwise, we will start a new neighbour discovery process by increasing its power gradually. The recovery stops when it finds another forwarding choice with a valid route cached toward one of the sinks. It is assumed that data acknowledgements are sent at the link layer. When a node does not receive an acknowledgement after a certain time, we assume the downstream link is invalid and then the routing is reconfigured.

V. ANALYSIS

Lemma1. The routing graph of the WSN is loop-free.

Proof: Suppose that there exists a loop A→B→C→D→E→⋯→A. Each node selects the next node which has lower height toward the sink. When a node is stuck in a local minimum, the node could increase its transmission range to find another node that has lower height toward the sink if it exists. Therefore, *height* (A) < ⋯ < *height* (E) < *height* (D) < *height* (C) < *height* (B) < *height* (A). This is a contradiction. □

Theorem1. If a route from a node to the sink exists, RTRR can find a route toward the sink.

Proof: From Lemma 1, we know that there is no loop in the routing graph. Since the number and height of nodes is limited, the route will lead to the sink as long as the real-time route exists. □

Theorem2. If the real-time route from a node to the sink exists, RTRR can find such a route.

Proof: We denote *delay* (*sink, i*) as the estimated delay; that is the average minimum delay from the sink to a node, while *delay* (*i, sink*) as the delay from the node to the sink on the counterpart route path. We denote T (*i, sink*) as the realistic delay experienced from the node to the sink.

The queuing delay T_q (*sink, i*) ≤ T_q (*i, sink*) and is bounded by the maximum queuing delay T_q (*i, sink*) ≤ T_{qmax}. When assuming the same radio and link quality for downstream and upstream links on the counterpart route, we get *delay* (*i, sink*) ≤ $delay_{qmax}$ (*sink, i*). $delay_{qmax}$ (*sink, i*) is the maximum queuing delay from the sink to node *i*. Then our estimated delay and realistic delay on route T satisfy that *delay* (*sink, i*) ≤ T ≤ $delay_{qmax}$ (*sink, i*). So, T_q (*sink, i*) ≤ T_q (*i, sink*). Also recall that when assuming same link quality, *delay* (*sink, i*) ≤ *delay* (*i, sink*) ≤ T (*i, sink*).

In RTRR we use *delay* (*sink, i*) to estimate delay from a node to the sink in routing discovery to find a route that meets the lower delay threshold, i.e., using *delay* (*sink, i*) to estimate T (*i, sink*). In this way, we can improve the real-time delivery ratio from nodes to the sink. Since we measure average delay with HEIGHT using power p_0, we get the maximum delay estimation *delay* (*sink, i*) on the minimum delay route from the sink to a node within different power levels. We find a relay node *i* where the delay T from *i* to the sink should be no more than the estimated delay on the route, i.e., T (*i, sink*) ≤ *delay* (*sink, i*). Otherwise, we increase the power level to find another forwarding choice *j*. Node *j* (with increasing power) must satisfy: *delay* (*sink, j*) + *Ave_delay* (*i, j*) ≤ T_{slack}; where T_{slack} = T_{max} − T (*s, i*). The end-to-end delay T must also satisfy: T (*s, sink*) = T (*s, i*) + T (*i, sink*) ≤ T (*s, i*) + *Ave_delay* (*i, j*) + *delay* (*sink, j*) ≤ T (*s, i*) + T_{slack} ≤ T_{max}. So, we find a route from node *s* to the sink that satisfies T (*s, sink*) ≤ T_{max}.

From the above, if a real-time route exists, our protocol can find a route satisfying that the end-to-end delay is within the delay requirement T_{max}. □

VI. SIMULATIONS

We verify our RTRR routing protocol using well-known ns2 simulator based on the parameters of MICAz motes as summarised in Table 1. All nodes have three power levels and the traffic pattern is many-to-one. In this simulation, we use a grid topology, which would be expected to conform to an in-building deployment. The network topology is shown in Fig. 2. We randomly select four nodes as source nodes and place one to four sinks (node 99, 98, 97 and 96) in the simulation area. Each source generates constant bit rate (CBR) traffic periodically. The real-time packet miss ratio is the ratio of all packets missed because of the delay bound to the total of packets sent. A fire breaks out 30 seconds after the simulation is started and in a random location. We use a fire model where fire spreads to its neighbours continuously every 10 seconds. When fire reaches a node, the node becomes unsafe after 10 seconds.

TABLE I. SIMULATION PARAMETERS

Parameter	**Value**
Propagation model	Shadowing
PhyType	Phy/WirelessPhy/802_15_4
MacType	Mac/802_15_4
CSThresh_ (carrier sense threshold)	5.29754e-11
RXThresh_ (receive threshold)	5.29754e-11
Pt_(transmit power)	5.35395e-05 / 0.000214158 / 0.000481855
Freq_	2.4e+9
Traffic	CBR
Traffic packetSize_	70
Traffic Interval_	0.0969
Node Initial energy	3.6 J

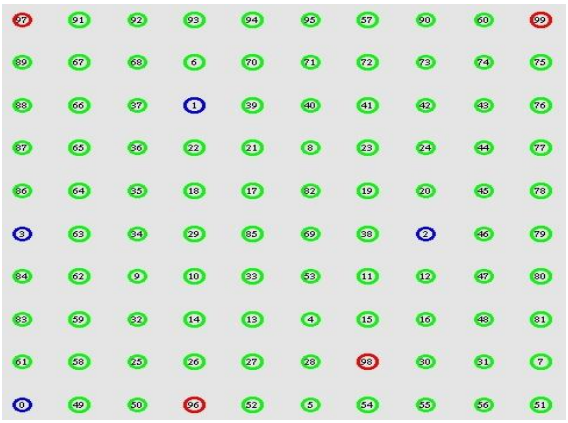

Figure 2. Simulation grid

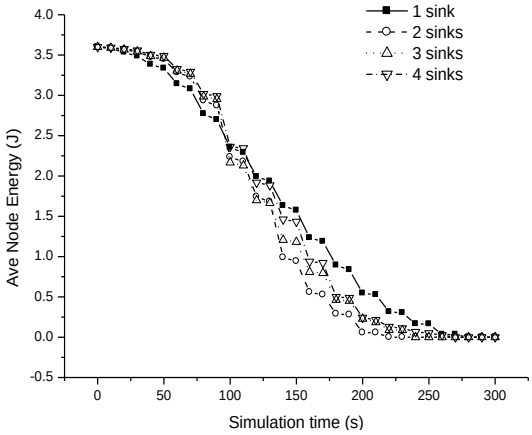

Figure 5. Average node energy when delay bound = 70 ms

Fig. 3 shows the end-to-end delay as the delay bound increases. The end-to-end delay decreases as the number of sink increases, because more sinks incur more packet delivery within the bound. Fig. 4 shows the miss ratio when the delay bound increases. The miss ratio decreases as the number of sink increases from one to four. Fig. 5 shows nodes' average residual energy in the simulation until the 300th second when the delay bound is 70 ms. The average energy does not vary greatly when the number of sink increases, as more sinks result in more packets are delivered and less routing trials with increased power are performed.

Fig. 6 illustrates the end-to-end delay with and without power adaptation using one and three sinks. Fig. 7 shows the miss ratio with and without power adaptation. The miss ratio rises greatly if we adapt the power level to increase the probability of real-time packet delivery. Fig. 8 illustrates the average energy in the simulation when the delay bound is set to 50 ms. Fig. 9 shows the miss ratio of real-time packet delivery with one sink. While RTRR achieves the best real-time delivery, RPAR [4] is not suitable for fire. Because, even though it can adapt its power level to find a real-time delivery path, its performance is bad in fire situations.

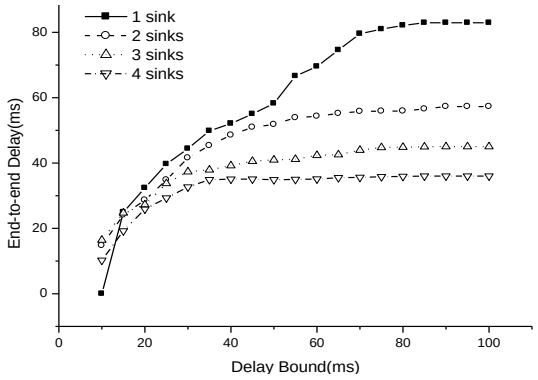

Figure 3. End-to-end delay as delay bound increases

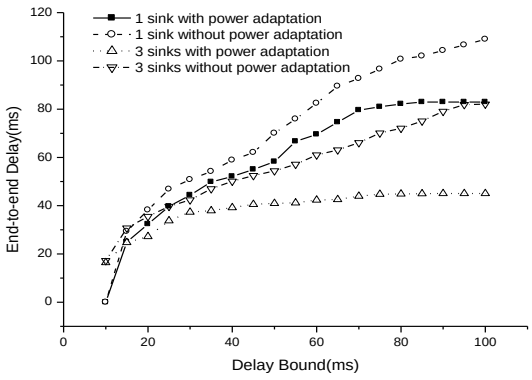

Figure 6. End-to-end delay with and without power adaptation

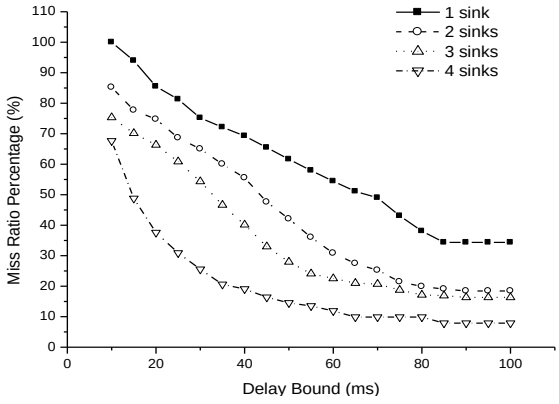

Figure 4. Miss ratio percentage as delay bound increases

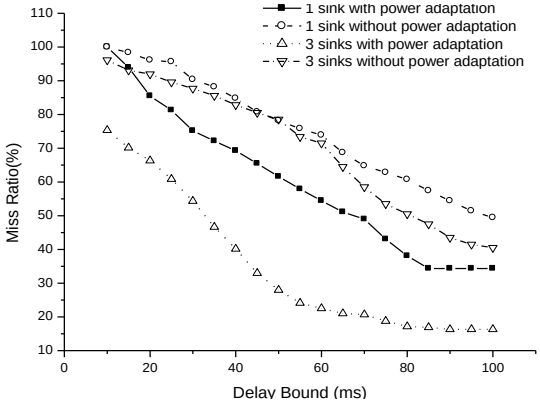

Figure 7. Miss ratio with and without power adaptation

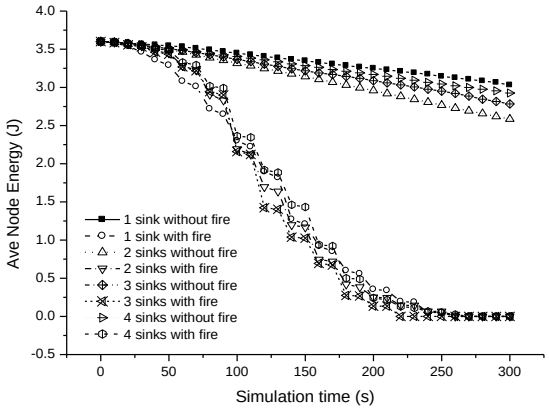

Figure 8. Average node energy when delay bound = 50 ms

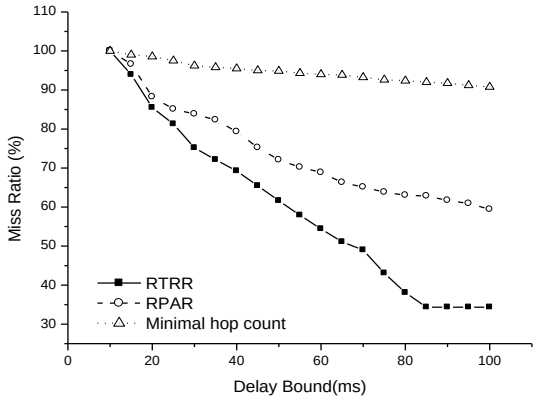

Figure 9. Miss ratio percentage as delay bound increases

Fig. 10 shows the average node energy in the simulation when the delay bound is 50 ms. The three routing protocols compared in this simulation have similar energy efficiency. RTRR increases its power level in order to increase real-time packet delivery, but it consumes more energy.

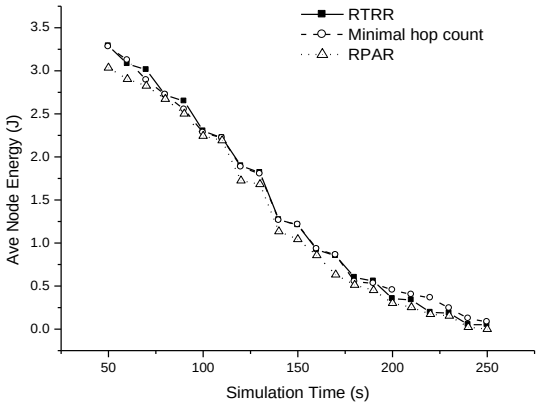

Figure 10. Average node energy when delay bound = 50 ms

VII. Conclusions and Future Work

We present a novel real-time and robust routing mechanism that is designed specifically for emergency applications such as building fire. The probability of real-time data delivery is achieved by maintaining a desired delay based on message propagation estimate and power level adaptation. The design is adaptive to realistic application characteristics including fire expanding, shrinking and diminishing. Our routing mechanism is designed as a localised protocol that makes decisions based solely on one-hop neighbourhood information. The simulation results prove that RTRR achieves good real-time packet delivery in fire situation when compared with other related protocols. We have recently validated our protocol on a 4-node TinyOS testbed and will in future deploy on a 20-node testbed for building fire response experiments.

References

[1] Networked Embedded Systems (NEMBES), http://www.nembes.org.

[2] T. He, J. Stankovic, C. Lu, and T. Abdelzaher, "SPEED: A Stateless Protocol for Real-time Communication in Sensor Networks," *ICDCS'03*, 2003.

[3] E. Felemban, C.–G. Lee, E. Ekici, R. Boder, and S. Vural, "Probabilistic QoS Guarantee in Reliability and Timeliness Domains in Wireless Sensor Networks," *IEEE InfoCom'05*, 2005.

[4] O. Chipara, Z. He, G. Xing, Q. Chen, *et. al.*, "Real-time power-aware routing in sensor networks," *14th IEEE International Workshop on Quality of Service*, 2006.

[5] A. Ahmed and N. Fisal, "A real-time routing protocol with load distribution in wireless sensor networks," *Computer Communications*, 31(14), pp.3190-3203, 2008.

[6] B.-L. Wenning, D. Pesch, A. Timm-Giel, and C. Gorg, "Environmental monitoring aware routing: making environmental sensor networks more robust," *Telecommunication Systems*, 2009.

[7] Y.-C. Tseng, M.-S. Pan, and Y.-Y. Tsai, "Wireless sensor networks for emergency navigation," *IEEE Computer*, 39(7), pp. 55-62, 2006.

[8] M.-S. Pan, C.-H. Tsai, and Y.-C. Tseng, "Emergency guiding and monitoring applications in indoor 3D environments by wireless sensor networks," *Int. J. of Sensor Networks*, 1(2), pp. 2-10, 2006.

[9] M. Barnes, H. Leather, and D. K. Arvind, "Emergency evacuation using wireless sensor networks," *32nd IEEE Conference on Local Computer Networks(LCN)*, 2007.

[10] J. Deng, R. Han, and S. Mishra, "A robust and light-weight routing mechanism for wireless sensor networks," *1st Workshop on Dependability Issue in Wireless Ad hoc networks and Sensor Networks*, 2004.

[11] T. Tabirca, K. Brown, C.J. Sreenan. "A Dynamic Model for Fire Emergency Evacuation Based on Wireless Sensor Networks," *8th International Symposium on Parallel and Distributed Computing (ISPDC)*, 2009.