A FIRE MONITORING APPLICATION FOR SCATTERED WIRELESS SENSOR NETWORKS

A Peer-to-Peer Cross-layering Approach

Luis Bernardo, Rodolfo Oliveira, Ricardo Tiago and Paulo Pinto
Faculdade de Ciências e Tecnologia, Universidade Nova de Lisboa, P-2829-516 Caparica, Portugal

Keywords: Alarm Application, Wireless Sensor Networks, Critical Application, Peer-to-Peer.

Abstract: A cross-layering alarm application is proposed for supporting fire fighting operations. It runs on scattered wireless sensor networks (WSN) composed by several isolated WSNs, where sensor nodes can be destroyed by fire. Mobile patrol nodes deploy the alarm monitoring application and collect the alarm records, containing the set of sensor measurements above the threshold values. The application was implemented in TinyOS 2.0, on Telos B motes. It uses a new Multimode Hybrid MAC, which can be controlled by the application. The application uses asynchronous mode when no alarms are active to optimise energy consumption; changes to full on mode (without sleeping) to minimise delay during fire handling situations; and uses the synchronous mode (with reserved bandwidth) during the transference of alarm records to the patrol node, balancing delay and energy saving. The alarm application organises sensor nodes into a clustered virtual overlay network and run a peer-to-peer searching service on top of it. This service is used to locate nodes outside the danger area, and to locate alarm records. The application performance was tested using TOSSIM simulations. Simulations results show the application capacity to capture a fire evolution.

1 INTRODUCTION

Wireless sensor networks (WSN) are likely to become widely deployed in the future, when it is technologically and economically feasible to produce small and low-cost sensors. Meanwhile, off-the-shelf experimental platforms like the Telos B (Polastre, 05) motes allow us to start developing WSN's future applications. This paper presents an alarm application based on Telos B motes to help fire fighting operations. Using the integrated temperature, light and humidity sensors, motes can monitor the fire favourable conditions in difficult access environments. Telos B motes' radio can reach up to 100m outdoor, and up to 30m indoor, but its radio range is usually lower. For covering a large area (e.g. Peneda-Gerês National Park in Portugal, with 72000 ha) a very large number of motes would be needed to create a continuous and dense WSN. For wide range coverage, this assumption, although considered reasonable in some WSN systems, may not be feasible (Meguerdichian, 01). A WSN for fire fighting was proposed and tested in (Hartung, 06). However, specially made motes were used, with long range directional radios and long range web camera sensors, which reduce drastically the total number of sensors needed to cover a vast region.

On this paper we assume that isolated islands of fixed randomly deployed motes exist, forming several scattered WSNs. Mobile patrol nodes (laptops or PDAs (Personal Digital Assistants) mounted on motor vehicles or fireman suits) roam the fields, connecting to the scattered WSNs. They control the application deployment, defining the sensor sampling rate and two alarm threshold conditions: a yellow threshold condition, for possible fire; and a red threshold condition, for imminent fire, where the possibility of a mote being destroyed is high. Motes store the alarm triggering sensor readings on other motes located outside the endangered area (when that is possible), improving resilience to fire. Patrol nodes can collect the stored alarms from the motes in real-time (e.g. receiving real-time warnings about fire enclosure danger), or afterwards, with the fire propagation history.

The next section presents an overview of the application design. Section 3 presents the initial alarm deployment protocol. Section 4 presents the p2p searching service, and describes how motes handle alarm conditions. A description of how alarms are recovered by the patrol node is addressed in Section 5. Section 6 presents a set of simulation results focused mainly on how the application behaves during a fire. Finally, Section 7 draws some conclusions and presents future work.

2 APPLICATION DESIGN

The alarm application design has three main objectives:

- Energy efficiency, to improve mote lifetime;
- Low delay, to detect and store information about danger spots and fires in progress;
- Resilience to mote loss.

Energy efficiency is achieved by configuring the MAC and transport layer protocols accordingly to the alarm level, in a cross-layering approach. The multimode hybrid medium access control protocol (MH-MAC) (Bernardo, 07) was used. It can operate in an asynchronous low duty cycle mode when no alarms exist, minimizing energy consumption; and it can operate in a synchronous mode (with reserved slots) or in an always on mode (without sleeping), during yellow and red alarm conditions, minimising the delay at the MAC layer. No packets are exchanged when no alarms are active, and clocks drift freely. Since we assume a low mote density, we decided not to use the "frisbee" model (Cerpa, 01) where neighbour motes run coordinated deviated low frequency sensor scans.

Energy efficiency and fire resilience is also achieved by optimising the alarm record storage on the WSN network. A fast lookup operation is needed during alarms to search for a suitable mote to store the records, or to move previously stored records outside the risk area during an alarm situation. An energy efficient lookup operation is also needed to support alarm record lookup for patrol nodes. During initial deployment, the alarm application creates a virtual overlay network (VON) composed by a minimum number of motes that fully connect all motes on a WSN island. This set is usually called a minimum connected dominant set (MCDS) (Wu, 99). The motes outside the MCDS are connected to one or more motes in the MCDS, defining clusters. Alarms are stored on MCDS members. The motes within the MCDS run a peer-to-peer (p2p) protocol similar to Gnutella (Chawathe, 03), to support mote and alarm record search. A caching mechanism was added to reduce the search overhead, and to favour alarm record grouping. MCDS members act as ultra-peers, conducting searches on their cluster members' behalf. Some authors (Greenstein, 03) proposed the use of distributed hash table based p2p approaches for locating information on WSNs. However, although motes are static they may fail silently due to battery exhaustion or destruction by fire, when the lookup operations are most needed. (Bernardo, 04) shows that DHT approaches may fail for these conditions, and that a flooding based p2p approach enhanced with a proper caching mechanism handles topology changes is a safer way. The trade-off is shown in (Liu, 07), where the cost of pushing index information is only effective when the number of searches is high enough. Otherwise, it is better to rely on a search based approach.

The alarm application has three different phases: the initial deployment; the alarm handling phase; and the alarm collecting phase, which may overlap in time with the alarm handling phase. The next three sections present these phases thoroughly.

3 ALARM DEPLOYMENT

When motes are initially deployed in the field, they run a low duty-cycle idle application. MH-MAC is set to asynchronous mode, and periodically turns-on the radio listening for incoming packets. It is assumed that all motes use the same duty cycle period ($\boldsymbol{T_{DUTY\_CYCLE}}$), equal to 1.1 seconds in the current prototype.

A moving patrol node looking for a WSN periodically broadcasts a *WSN-DISCOVERY* packet using MH-MAC asynchronous mode. A sequence of small preambles with a total duration of $2T_{DUTY\_CYCLE}$ is sent before the *WSN-DISCOVERY* packet, to guarantee that all receivers are awake (see Fig. 1). After receiving the packet, receivers run a random backoff timer and send a *WSN-ACK* packet, signalling their presence and their energy level. The patrol node selects one proxy mote using the RSSI (*Received Signal Strength Indication*) measured during the packet reception, and the energy level reported by the mote. It selects the mote with the highest RSSI level, and the energy value above a minimum threshold value.

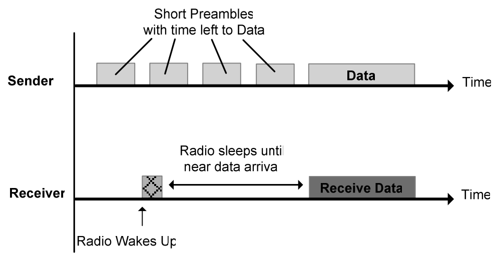
Short Preambles with time left to Data
Sender
Data
Time
Radio sleeps until near data arrival
Receiver
Receive Data
Time
Radio Wakes Up

Figure 1: MH-MAC asynchronous broadcast transmission.

During an initial alarm deployment, the proxy mote starts the flooding of an *ALARM-SET* packet. This flooding simultaneously activates a sensor monitoring thread on the motes, and defines a MCDS tree. MH-MAC asynchronous mode is used. The *ALARM-SET* packet contains a decrementing hop counter for controlling the area where the alarm is activated, and two numeric expressions for the yellow and red threshold alarm levels. The receiver motes re-flood the *ALARM-SET* decrementing the hop counter, until it reaches zero. Re-flooding collisions are reduced by running a slotted random backoff at each receiver mote. Receiver motes select one of ten slots to re-broadcast the *ALARM-SET* packet, counted from the end of the *ALARM-SET* transmission. Each slot has a duration of

$$\boldsymbol{T_{SLOT}= 2T_{DUTY\_CYCLE} + \alpha}, \tag{1}$$

where α= 0.1 seconds represents the time for a short initial random backoff time (up to 30 ms), and the *ALARM-SET* packet transmission time. If a preamble transmission is detected during the short backoff time before the packet transmission, the mote cancels its transmission in this slot and selects a new one. MH-MAC can handle packet collisions when two or more transmitters are outside radio range but their radio coverage areas intercept. Receiver motes send a *SHUT-UP* packet with a configurable probability when a collision is detected. This packet includes the active sender address and signals other senders that a transmission is in progress. Broadcast receivers wake up a few milliseconds before the data arrival to detect preamble collisions.

A mote assumes that all neighbours already retransmitted an *ALARM-SET* packet when it does not receive any retransmission during at least a T_{IDLE} time (equal to 45 seconds in the prototype). The mote then starts the MCDS creation phase, running the algorithm presented in (Wu, 99). Initially, it broadcasts a *NEIGHBOUR-TABLE* packet using MH-MAC asynchronous mode, and waits for the reception of the same packet from all its neighbours. The *NEIGHBOUR-TABLE* packet contains the list of neighbours, and allows the motes to know their neighbourhood within two hops range. Each mote decides if it is a cluster head (CH) using the following rules (Wu, 99): it initially self-selects as CH if two unconnected neighbours exist; afterwards, in a second phase, it excludes itself from CH role if another mote, or two motes, with the same number or more of neighbours and with higher addresses exist. All motes that self-select as CH, broadcast a *GATEWAY* packet and pre-allocate a buffer in the flash (64Kbytes) for storing alarm records. Motes maintain a list of all neighbour CH motes.

The alarm deployment protocol is illustrated in Fig. 2. The proxy mote *A* broadcasts the *ALARM-SET* packet, which is re-broadcasted by motes *B* and *C* in two different slots. Afterwards, they exchange the *NEIGHBOUR-TABLE* packets, and mote *A* self-selects as a CH.

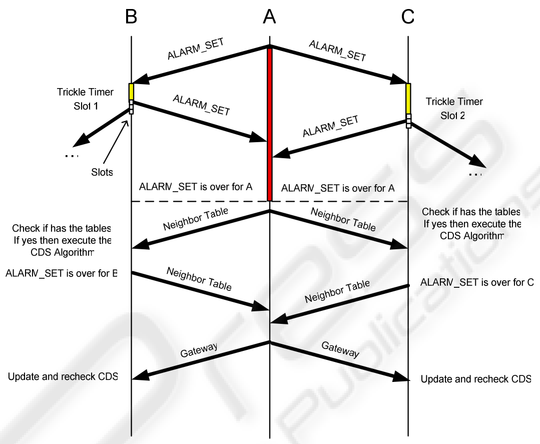

Figure 2: *ALARM-SET* flooding protocol example.

The minimum duration of an alarm deployment phase can be calculated for a regular network with a depth of *r* hops, where motes have a fixed number of *K* neighbours, using (2). It assumes that only *K-1* slots are used per *ALARM-SET* sent by the neighbours (except for the proxy mote that uses *K* slots). It also assumes that the descendent motes do not interfere in their transmissions and that after *K-1* slots plus the idle time all descendent motes transmit the *NEIGHBOUR-TABLE* packet.

$$\begin{aligned}&DeploymentTime \geq \\ &(r-1)((2K-1)T_{SLOT} + T_{IDLE}) + 2T_{SLOT} + T_{IDLE}\end{aligned} \tag{2}$$

The alarm deployment operation can have a large duration. However, this operation does not need the support of the patrol node. After sending the *ALARM-SET* packet to the proxy mote, the patrol node can leave the WSN region. It also, does not translate into a large energy cost, because each packet transmits at most three packets (possibly more if collisions occur), and MH-MAC is very aggressive putting motes into sleep during broadcast packet receptions. Finally, this protocol is run outside a critical period. It creates a backbone network for supporting the alarm handling protocol, run during the critical periods.

4 ALARM HANDLING

After the alarm application deployment, motes run autonomously a sensor scanning thread, maintaining MH-MAC in the asynchronous mode to save energy. No VON maintenance is done during the long idle periods that can exist before an alarm is detected. Periodically (by default, every 20 seconds) motes test the sensor values for the red and yellow threshold conditions.

When an alarm state is detected, motes change to an alarm handling operation mode, where: MH-MAC is set to full-on, or synchronous mode; the sensor sampling rate is increased (by default, to 5 seconds); and the sensor readings are stored in a repository mote, selected from the CH motes located in a safer zone. The MH-MAC mode transition is critical to the overall performance of the alarm application, due to the large time overhead per packet on the MH-MAC asynchronous mode. It is important to use conservative yellow alarm threshold values to allow the mode transition to occur before the information transfer peak. However, the cost of false alarms must also be taken into account. A key component to the overall performance is the searching service.

4.1 Peer-to-Peer Searching Service

CH motes run a peer-to-peer (p2p) searching service to support the discovery of motes outside the alarm zone, and the discovery of alarm records. The MCDS defined during the deployment phase fully connects all motes. Therefore, it constitutes a virtual overlay network (VON) of CH motes, which can efficiently run a flooding based search, like the one proposed in Gnutella. Its performance was enhanced applying a caching mechanism inspired on AODV (Ad-hoc On-demand Distance Vector) routing protocol (Perkins, 03).

The basic search mechanism is the flooding of a *QUERY* packet over the MCDS VON. The *QUERY* packet includes a hop counter and a query definition part, which only has two values for the alarm application: "search for CH in no alarm state"; or "search for alarm record". CH motes receiving a new *QUERY* packet with a nonzero hop counter decrement it, and flood the packet over all CDS links except the one where it came from. They also store a QUERY forwarding record to detect duplicate packets, route future *HIT* packets, and to maintain a cache of *HIT* packets. After receiving the *QUERY* packets, CH motes check their local state (alarm state and the repository of alarm records) and their *HIT* local cache. They send the *QUERY* packet issuer a *HIT* packet with the results. The *HIT* packet includes the sender's alarm state, the path alarm state (the highest alarm state of a CH in the path), and the full path to the sender mote (for routing purposes).

An expanding ring search approach is used to reduce the flooding search cost (Chawathe, 03). The search for an idle alarm (green) CH stops when a green CH is found, or when the maximum range specified by the application is reached. In this case, a yellow alarm mote is selected if it exists, with a path that crosses preferentially yellow motes.

The *HIT* caching mechanism reduces the searching overhead, creating a client driven index distribution through the VON. Since a fire evolution forecast is not easy due to unpredictable factors (e.g. wind), it is not easy to push HIT packets before they are actually needed as (Liu,07)(Lee,06) propose. On the other hand, unpredictability also introduces the requirement for an efficient *HIT* cache invalidation mechanism. The AODV's active neighbour table approach is used. CH motes maintain a table of CH neighbours to whom an *HIT* packet was sent (originated locally, or forwarded). When their alarm state changes (an alarm threshold is reached), a new *HIT* packet is sent to all motes in the active neighbour table, updating the previous cached packets. Remote cached *HITs* are also updated. They are sent when an intermediate node changes it alarm state modifying the path alarm level. Finally, cancelling *HIT* packets are sent when a mote failure is detected.

The MCDS VON is not maintained during the idle periods, and can suffer from CH lost by battery exhaustion, or during a fire. Therefore, the search protocol was enhanced to restore the VON in response to mote losses. *QUERY* and *HIT* packets are sent using unicast. They can be used to detected link loss because they are acknowledged, contrarily to the broadcast messages. If a packet transmission fails for more than three tries, then the service assumes that a link is lost. Using the previously stored information, the mote runs the CH self-selection algorithm, and sends the *GATEWAY* packet if it sets himself as CH. If the red alarm threshold is not active, the *NEIGHBOUR-TABLE* is also sent. Non red alarm motes do not consider neighbour red alarm motes during the MCDS creation to improve the VON survivability. The MCDS is also updated when a new mote is discovered, due to overhearing its packets.

Fig. 3 illustrates the search algorithm usage. After running the alarm deployment algorithm, motes *B*, *C*, and *D* are selected as CH. Mote *A* starts a two hop search for an idle CH (in the Green zone), sending a *QUERY* packet to its neighbour CH (mote *B*). The *QUERY* packet is forwarded to CH *C* and *D*, and mote *A* receives *HIT* packets from both CHs, reporting their alarm states and if they have any alarm record stored (preloading the caches for future searches). All CHs store in their cache the *HIT* packets received. CH *D* adds *C* to its active neighbour table, and CH *C* does the same for CH *B*. If the fire spreads and mote *C* is destroyed or reaches the red alarm level (and *B* and *D* remain active), then motes *F* and *G* change their role to CH, updating the VON on demand. CH *B* deletes from its cache the *HIT* packets received from *C* and *D* when it detects that mote *C* failed.

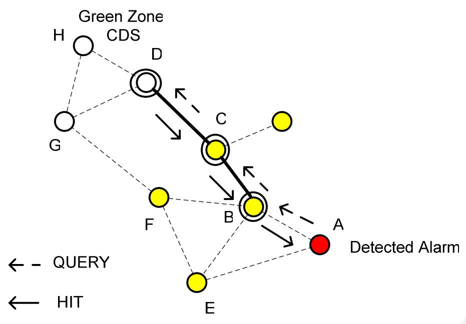

Figure 3: A search on the mote VON.

4.2 Alarm Information Storage

When a mote detects an alarm condition, it increases its sampling rate and starts collecting the local sensor alarm information. In order to reduce the amount of information stored for each alarm, nodes only record changes on the sensor values above a minimum deviation value. The mote uses the searching service presented above to select a green CH. CHs only generates a local HIT if they still have flash space to store extra alarm records. Using the path contained in the *HIT* packet, the mote can send *ALARM-RECORD* packets to the selected CH. These packets include a mote identifier, a sequence number, a time value, and can group several individual sensor changes. If a CH receiving *ALARM-RECORD* packets also detects an alarm condition, then it also uses the searching service to locate a safer CH to store the records. In this case the transfer of the previous stored *ALARM-RECORD* packets to the new CH can generate a large peak of traffic. The PSFQ (Pump slowly, Fetch Quickly) reliable transport protocol (Wan, 02) is used to improve the *ALARM-RECORD* transference reliability. *ALARM-RECORD* packets are locally confirmed at each hop, and CHs generate *ALARM-NACK* packets when a loss is detected.

The *ALARM-RECORD* transference time depends strongly on the MH-MAC mode. Unicast packet transmission in asynchronous mode is more reliable and faster than broadcast transmission. The preamble ends when the sender receives an early acknowledgment. (Bernardo, 07) compares the performance of the three MH-MAC modes and shows that asynchronous mode is only effective for up to two senders. Above that, only synchronous mode and full-on mode are capable of supporting acceptable throughput and delay. Full-on mode minimises delay (less than 100 ms for ten concurrent sending motes) at the cost of more collisions and energy consumption. Synchronous mode maximises throughput and minimises energy consumption for more than four sending motes, at the cost of extra delay (about 500 milliseconds). The alarm application can have three different configurations: *FullOn*; *FullOn2Demand*; *Synchronous*.

On the *FullOn* configuration, all motes change its mode to *Full-on* before a fire occurs. A patrol node command can be used to activate the change. Unfortunately, this model is not always possible due to the unpredictability of fires and it is used for benchmarking the other approaches.

On the *FullOn2Demand* configuration, motes only change to the *Full-on* mode when a local alarm condition is detected. CH motes also change to *Full-on* mode when a mote in their cluster is in alarm state, or when they receive an *ALARM-RECORD* packet. This means that they are being used to transfer *ALARM-RECORD* packets between CHs, or from a CH to a patrol node. CH motes return to the asynchronous mode when no *ALARM-RECORD* packets are received for more than IDLE-TX seconds (by default 60 seconds), if the change was triggered only by a packet flow.

The third scenario assumes that motes stay always in a synchronous mode, including during the idle periods. This scenario is usually assumed in periodical data collection works based on periodical synchronous MAC protocols (e.g. (Cerpa, 01)). However, it requires the clock synchronization overhead during the idle periods.

5 ALARM RECOVERING

When the patrol node wants to collect alarm records, it runs the *WSN-DISCOVERY* packet broadcast protocol presented in section 3 to discover the WSN and to select a proxy mote. The proxy selection rule now also takes into account if a mote is CH. The patrol node sends the mote a *QUERY* packet, requesting all alarm records within a user specified range. Non-CH motes send the request to a CH, and forward the received *HIT* packets to the patrol node.

The *ALARM-RECOVER* packet is used to request the stored *ALARM-RECORDs*. It includes the alarm application identifier field (defined during the initial deployment), and the start and the stop time fields, which define a time limit for the alarms to collect. Future stop times request a continuous feed of *ALARM-RECORD* packets to the patrol node until the stop time is reached.

Initially, during the access to the searching service only the proxy mote changes its MH-MAC mode to synchronous, and synchronises with the patrol node. Afterwards, all CH motes in the path to CH motes storing *ALARM-RECORD*s also change their mode to synchronous, and dedicated channels are reserved connecting the source CHs to the patrol node. The *ALARM-RECOVER* packets are source routed using the path received in the *HIT* packets. When a CH receives this packet, it changes its MH-MAC mode to synchronous (if it is not already there) and requests MH-MAC to synchronize with the next CH in the path.

6 PERFORMANCE EVALUATION

The alarm application prototype was implemented in TinyOS 2.0 (TinyOS2.0, 07) and was tested on Crossbow Telos B motes. However, due to the small number of motes available for this project, the performance evaluation was done using the TOSSIM simulator (Levis, 03). The current TOSSIM version does not support the CC2420 radio stack used by the LPL (low power listening) library and the flash access. Therefore, we emulate the CC2420 radio stack and modified TOSSIM interface implementations to simulate the flash access time. Additionally, meters were placed on the MAC and the application code to measure the number of milliseconds used for data transmissions, for data receptions, for flash reads and writes, and the time spent in active and radio sleep states. Using the current consumption specifications shown in Table 1, we were able to estimate the total current consumption for the tested scenarios. We considered that in idle or receiving state the mote has the consumption of operation MCU+Radio RX, in radio sleep it has the consumption of operation MCU Active, and during packet transmissions it has the consumption of operation MCU+Radio TX.

Table 1: Telos B current consumption (Polastre, 05).

Operation	Current
Mote Standby (RTC on)	5.1μA
MCU Idle (DCO on)	54.5μA
MCU Active	1.8 mA
MCU + Radio RX	21.8 mA
MCU + Radio TX (0dBm)	19.5 mA
MCU + Flash Read	4.1 mA
MCU + Flash Write	15.1 mA

The performance results presented in this paper focus on the alarm handling phase. We simulated the WSN presented in Fig. 4 with 55 fixed motes, with an average link distance of 20 meters. The yellow alarm and the red alarm expressions were respectively "*SOt* > 40ºC" and "*SOt* > 60ºC", where *SOt* is the sensor output temperature value. The temperature increases at a fixed rate of 10ºC per minute after the yellow alarm is set, and the motes fail when the temperature reaches 80ºC.

Initially, all motes are idle (no alarms are active). Then, a head of fire crosses the WSN following the arrow direction, at a variable speed. When the head of fire reaches the motes represented in red (1, 2, etc.) they trigger the yellow alarm, 120 seconds later they trigger the red alarm, and after 120 seconds more the motes are destroyed. Motes represented in yellow (3, 4, etc.), temporarily trigger the yellow alarm, during an interval of 360 seconds. The remaining motes (6, 7, etc.) are not affected by the fire. 72 *ALARM-RECORD* packets are generated for each red mote, and 48 *ALARM-RECORD* packets are generated for each yellow mote. Seven head of fire speeds were simulated, ranging from 50 meters/hour to 1000 meters/hour.

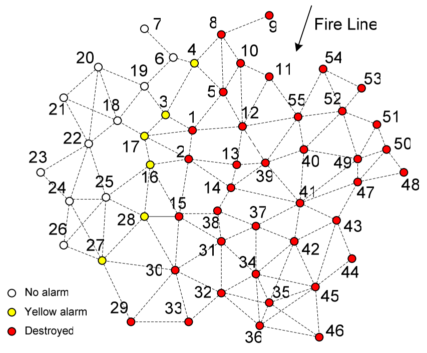

Figure 4: Simulated WSN.

This section analyses the performance of *FullOn*, *FullOn2Demand*, and *Synchronous* configurations, presented in section 4.2. The asynchronous MH-MAC mode used a duty cycle period (T_{DUTY_CYCLE}) of 1.1 seconds, and a duty cycle of 9%. Packet transmission is preceded by a sequence of preambles lasting 2.2 seconds for broadcast, and an average of 0.55 seconds for unicast packets due to the early acknowledgement mechanism. Synchronous mode uses a periodic slot structure with 11 slots of 100ms. Communication between motes is all done through reserved slot, guarantying that more than 10 messages can be exchanged between motes for each period without being lost in collisions. However, MH-MAC organizes the slots in a single ladder, minimising the delay only in one direction. The average per hop delay is 0.55 seconds.

Fig. 5 and Fig. 6 show the percentage of *ALARM-RECORD* packets that were stored in the surviving motes, respectively when a pure flooding approach is used, and when *HIT* caching and *HIT* invalidation is used. Fig. 5 shows that a simple flooding search approach fails to cope with the fire dynamics. The application starts loosing records in all three approaches for a head of fire speed of 50 m/hour, equivalent to 2.5 WSN hops per hour. The most critical operation is the search for a safe CH to transfer locally stored *ALARM-RECORDs*. Whenever the existing path fails, a *QUERY* is flooded to locate a new safe path to a safe CH. For CH motes located far from the green zone (e.g. mote 45) require a four hop *QUERY* packet flooding, each time an intermediate mote fails. When the head of fire speed is increased more *ALARM-RECORDS* are lost, mainly due to CHs that do not transfer stored alarm records, or became isolated (e.g. motes *36* and *46*). The *ALARM*-RECORD survivability is severely affected by the hop-by-hop delay, and *Synchronous* configuration has the highest delay value, followed by the *FullOn2Demand* configuration.

Fig. 6 shows that the proposed HIT caching and HIT invalidation methods are capable of effectively improving the *ALARM-RECORD* survival rate for all three configurations. They reduce the dependency on the hop-by-hop delay because the reaction to the approximation of the head of fire is anticipated. A new path is selected as soon as a mote in the path changes its alarm state. *HIT* updates maintain the validity of the cached *HITS*, and *QUERY* flooding only occurs when a longer path is needed. Therefore, motes have more time to transfer their records to a safe CH. Records are lost for head of fire speeds above 300 m/hour, equivalent to 15 WSN hops per hour. Fig. 6 results suggest that the dominant parameter is the available bandwidth to handle the record transference during the head of fire approach. The *Synchronous* MH-MAC mode provides more bandwidth for load peaks than the *FullOn* mode due to collision avoidance (Bernardo, 07). Fig. 6 also shows that the motes can change from asynchronous mode to full-on mode on demand, with a marginal loss in the alarm record survival rate. For the relative fastest speed (50 hops / hour or 1000 meters / hour), the alarm system was able to maintain more than 75% of the *ALARM-RECORD* packets generated during a fire for the *FullOn2Demand* approach, only 5% below the *FullOn* approach results. The *FullOn2Demand* approach could be improved if T_{DUTY_CYCLE} is shorter. This parameter affects all unicast and broadcast asynchronous transmission times, and indirectly the time it takes to change from the asynchronous mode to the Full-on mode. However, energy consumption on the idle periods would also be increased.

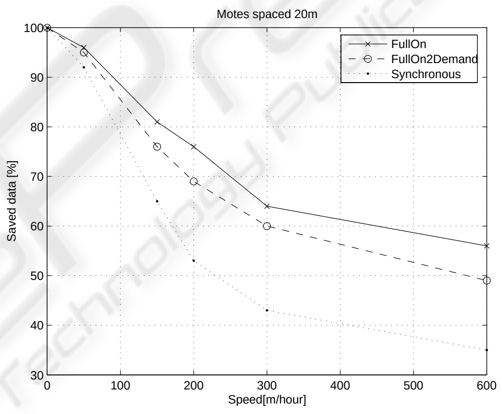

Figure 5: Influence of the fire front velocity in the percentage of saved data, without *HIT* invalidation.

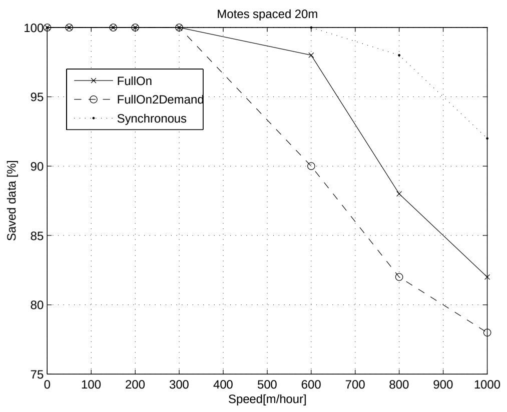

Figure 6: Influence of the fire front velocity in the percentage of saved data, with HIT invalidation.

The head of fire speed has no significant influence in the current consumption per mote and per second.

The average and standard deviation current consumed by the surviving motes (with yellow alarm or with no alarm) was 21.80 ± 0 mA, 10.62 ± 6.67 mA and 10.78 ± 2.23 mA, respectively for the *FullOn*, *FullOn2Demand*, and *Synchronous* configurations and 1000 m/hour. *FullOn2Demand* is clearly the less demanding approach if the energy savings during idle periods are taken into account.

7 CONCLUSIONS AND FUTURE WORK

This paper presents an alarm application designed to support fire fighting operations. It shows that it is possible to improve the mote longevity (avoiding the synchronization costs during idle periods) and still have a timely response to destructive events when the application controls the MAC behaviour. The reaction speed is mainly conditioned by the duty cycle period. In order to increment it, more energy must be spent in idle time.

Future work includes the test of an on-demand synchronous mode, the adaptation of the "frisbee" model to an asynchronous operation mode, and the thorough testing of the application on a large mote test bed to validate the simulation results.

ACKNOWLEDGEMENTS

This work was partially supported by the Fundação para a Ciência e Tecnologia under the project SIGAPANO POSC/EIA/62199/2004.

REFERENCES

Bernardo, L., and Pinto, P., 2004. A decentralized location service Applying P2P technology for picking replicas on replicated services. In *ICETE'04, 1st Int. Conf. on E-Business and Telecommunication Networks*, Vol.1 pp.39-47, INSTICC Press. Aug. 2004.

Bernardo, L., Oliveira, R., Pereira, M., Macedo, M., and Pinto, P., 2007. A Wireless Sensor MAC Protocol for bursty data traffic. In *IEEE PIMRC'07, 18th IEEE Annual International Symposium on Personal Indoor and Mobile Radio Communications*, Sep. 2007.

Cerpa, A., Elson, J., Estrin, D., Girod, L., Hamilton, M., and Zhao, J., 2001. Habitat Monitoring: Application Driver for Wireless Communications Technology. In *1st ACM SIGCOMM Workshop on Data Comm. in Latin America and the Caribbean*, ACM Press. Apr. 2001.

Chawathe, Y., et al., 2003. Making Gnutella-like P2P Systems Scalable. In *SIGCOMM'03, the 2003 conference on Applications, technologies, architectures, and protocols for computer communications*, ACM Press, Aug. 2003.

Greenstein, B., Estrin, D., Govindan, R., Ratnasamy, S., and Shenker, S., 2003. DIFS: A Distributed Index for Features in Sensor Networks. In *SNPA'03, 1st IEEE Workshop on Sensor Networks Protocols and Applications*, IEEE. May 2003.

Hartung, C., Han, R., Seielstad, C., and Holbrook, S., 2006. FireWxNet: A Multi-Tiered Portable Wireless System for Monitoring Weather Conditions in Wildland Fire Environments. In *MobiSys'06, 4th Int. Conf. on Mobile Systems, Applications, and Services*, pp.28-41, ACM Press. Jun. 2006.

Kim, H.S., Abdelzaher, T.F., and Kwon, W.H., 2003. Minimum-Energy Asynchronous Dissemination to Mobile Sinks in Wireless Sensor Networks. In *SenSys'03, 1st ACM Conf. on Embedded Networked Sensor Systems*, ACM Press, Nov. 2003.

Lee, U., Magistretti, E., Zhou, B., Gerla, M., Bellavista, P., and Corradi, A., 2006. Efficient Data Harvesting in Mobile Sensor Platforms. In *PerSeNS'06, 2nd IEEE Int. Workshop on Sensor Networks and Systems for Pervasive Computing*, IEEE, Mar. 2006.

Levis, P., Lee, N., Welsh, M., and Culler, D., 2003. TOSSIM: Accurate and Scalable Simulation of Entire TinyOS Applications. In SenSys'03, Int. Conf. on Embedded Networked Sensor Systems, ACM, pp. 126-137, Nov. 2003.

Liu, X., Huang, Q., and Zhang, Y., 2007. Balancing Push and Pull for Efficient Information Discovery in Large-Scale Sensor Networks. *IEEE Trans. on Mobile Computing* Vol. 6 No. 3, pp. 241-251, Mar. 2007.

Meguerdichian, S., Koushanfar, F., Potkonjak, M., and Srivastava, M., 2001. Coverage Problems in Wireless Ad-hoc Sensor Networks. In IEEE Infocom'01, 20th Annual Joint Conference of the IEEE Computer and Communications Societies, pp. 1380-1387, Abr. 2001.

Perkins, C., Belding-Royer, E., Das, S., 2003. Ad hoc On-Demand Distance Vector (AODV) Routing. IETF RFC 3561.

Polastre, J., Szewczyk, R., and Culler, D., 2005. Telos: Enabling Ultra-Low Power Wireless Research. In *IPSN'05, Int. Symp. on Information Processing in Sensor Networks*, IEEE, pp. 364- 369, Apr. 2005.

TinyOS2.0, 2007. TinyOS 2.0 Documentation. Retrieved Mar. 2007 from http://www.tinyos.net/tinyos-2.x/doc/

Wan, C.Y., Campbell, A.T., and Krishnamurthy, L., 2002. PSFQ: A Reliable Transport Protocol for Wireless Sensor Networks. In *WSNA'02, 1st Int. Workshop on Wireless Sensor Networks and Applications*, ACM, pp. 1-11, Sep. 2002.

Wu, J., and Li, H., 1999. On Calculating Connected Dominating Set for Efficient Routing in Ad Hoc Wireless Networks. In *DIALM '99, 3rd Int. Workshop on Discrete Algorithms and Methods for Mobile Computing and Communications*, ACM, pp. 7-14, Aug. 1999.