# BurstProbe: Debugging Time-Critical Data Delivery in Wireless Sensor Networks

James Brown[1], Ben McCarthy[1], Utz Roedig[1], Thiemo Voigt[2], and Cormac J. Sreenan[3]

[1] Lancaster University, UK
[2] Swedish Institute of Computer Science (SICS), Sweden
[3] University College Cork, Ireland

**Abstract.** In this paper we present *BurstProbe*, a new technique to accurately measure link burstiness in a wireless sensor network employed for time-critical data delivery. Measurement relies on shared probing slots that are embedded in the transmission schedule and used by nodes to assess link burstiness over time. The acquired link burstiness information can be stored in the node's flash memory and relied upon to diagnose transmission problems when missed deadlines occur. Thus, accurate diagnosis is achieved in a distributed manner and without the overhead of transmitting rich measurement data to a central collection point. For the purpose of evaluation we have implemented BurstProbe in the GinMAC WSN protocol and we are able to demonstrate it is an accurate tool to debug time-critical data delivery. In addition, we analyze the cost of implementing BurstProbe and investigate its effectiveness.

## 1 Introduction

Future application areas for wireless sensor networks (WSNs) are industrial process automation and control systems. In such systems, the WSN is part of a control loop, and therefore predictable network performance in terms of message transfer delay and reliability is required.

WSNs for such applications are generally built around a schedule-based Medium Access Control (MAC) protocol. The schedule is calculated such that deadlines are met even when some retransmissions are necessary to compensate for losses on wireless links. Due to deadline and/or energy constraints it is obviously not possible to accommodate an arbitrary number of retransmissions and therefore a worst-case link reliability has to be assumed when determining a transmission schedule. Recently developed systems for time-critical data delivery such as WirelessHART [3], ISA100 [4], Munir's least-burst-routing [7], and GinMAC [13] use worst-case reliability assumptions when determining schedules. In WirelessHART a fixed number of redundant transmissions is used in the hope that these are sufficient to compensate for losses. Munir's least-burst-routing and GinMAC determine worst-case link reliability by measurement before deployment and determine the number of required retransmission slots on links based on this measurement.

Recent deployments show [7, 13] that such provisioning before deployment is generally feasible, however, it cannot be guaranteed that link characteristics are invariant. For example, an interferer may appear (temporarily) in the vicinity of the network or links may become (temporarily) blocked by obstacles. It must be possible to either determine a new valid schedule or to identify and remove the problem source, which appeared in the sensor field post-deployment. It is necessary to collect appropriate debugging information during system operation allowing us to forensically investigate the problem that occurred.

It has been shown [7] that link quality in WSNs used for time-critical data delivery should be described with more precision than it is possible using simple metrics such as Packet Reception Rate (PRR) or Expected Transmission Count (ETX). To design efficient schedules for time-critical systems it is necessary to understand in detail the distribution and nature of burst errors on links. Unfortunately, as we show, it is impossible to gather such detailed link information by simply using existing data transmissions. Hence, it is very challenging to monitor link burstiness in a WSN deployment during network operation.

We propose to periodically measure link burstiness within the network in order to collect the necessary information for performance debugging in case that time-critical data delivery fails. Sequences of dedicated test transmissions - called BurstProbes - are used to obtain a clear picture of link burstiness over time. These probes are incorporated in the transmission schedule such that they do not interfere with the network's time-critical data delivery. As time-critical WSNs do not have sufficient spare capacity to continuously transmit all of their measurement data to a central collection point the data is stored in each node's flash memory. This data can then be retrieved during network maintenance after problems have occurred and network debugging is therefore required. The paper has the following specific contributions:

- BurstProbe: We introduce the novel concept of BurstProbe.
- BurstProbe Implementation: An implementation of BurstProbe within the GinMAC protocol and an experimental evaluation of its overhead and its effectiveness is presented.
- Debugging Examples: Different debugging examples using BurstProbe are presented. We show that problem sources can be identified and that new viable schedules can be calculated on the basis of recorded measurements.

We implement BurstProbe for GinMAC, a TDMA based MAC layer for time-critical data delivery that requires offline-dimensioning [13]. BurstProbe enables GinMAC to become adaptive by allowing it to determine the required number of retransmission slots when link characteristics change due to interference. BurstProbe is currently implemented in GinMAC, but it is suitable for use in any schedule-based WSN system aiming to achieve time-critical data delivery.

The paper is organised as follows. Section 2 introduces the BurstProbe concept. Section 3 provides a detailed description of the BurstProbe implementation within GinMAC. Section 4 investigates the effectiveness of BurstProbes via experimental evaluation. Section 5 reports on related work dealing with the deployment of time-critical sensor networks. Section 6 concludes the paper.

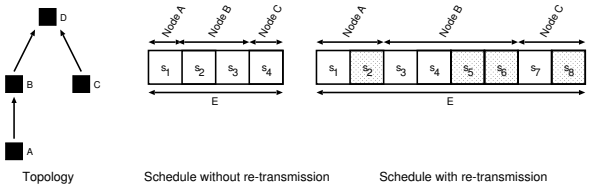



**Fig. 1.** Simple topology and possible schedules for error free and lossy channels.

## 2 BurstProbe

In this section we introduce the BurstProbe concept. We describe the motivation for its design and discuss its capabilities and limitations.

### 2.1 Scheduling for Timely and Reliable Data Delivery

Consider the simple network topology given in Fig. 1. Assume nodes $A$, $B$ and $C$ have to deliver data with period $T$ to the sink node $D$. In order to guarantee timeliness a TDMA schedule is applied. A slot in the schedule accommodates the actual data transmission and a short acknowledgment from the receiver. In this paper, transmissions within a slot refer to both the original data packet and corresponding acknowledgment. If we assume that all nodes are in interference range of each other, the schedule as shown in Fig. 1 can be used. Node $A$ transmits in slot $s_1$ to node $B$ which uses $s_2$ to forward data from $A$ to $D$; $B$ uses slot $s_3$ to transmit its own data to $D$. Node $C$ uses slot $s_4$ to transmit data to $D$. The resulting schedule $S = \{s_1, s_2, s_3, s_4\}$ has a duration (which we refer to as an *epoch*) of $E = |S| \cdot t = 4 \cdot t$ (with $t$ being the slot length). Data from all nodes is delivered within the epoch to the sink. We refer to the schedule as *valid schedule* if it allows us to deliver data within the required period $T$. The schedule is valid if the epoch is shorter than the period ($E \leq T$).

Obviously, the schedule is only valid in situations where all transmissions are successful. In a wireless environment error free channels are rare and capacity for potential retransmissions must be incorporated within the schedule. Figure 1 shows a schedule for the aforementioned simple topology which allows for one retransmission on each link for each transmission. The epoch length has now doubled to allow for reliable and timely data delivery on potentially lossy links. The schedule is valid if $E \leq T$ and if no more than every second transmission is erroneous. Given the harsh radio environment where some sensor networks operate it is a challenge to provision the correct number of retransmission slots in advance.

The epoch length can be reduced if all nodes are not within communication range of each other. In this case spatial re-use of TDMA slots is possible and the epoch can be shortened. However, it has to be noted that in industrial process automation and control scenarios in which time-critical scheduling is required it is common that all nodes are at least in interfering range of each

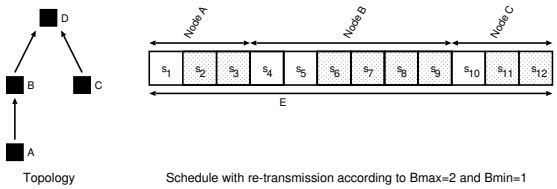



**Fig. 2.** Topology and possible schedules for lossy channels with $B_{max} = 2$, $B_{min} = 1$.

other. In practical industrial deployments spatial slot re-use is rarely an option and systems such as GinMAC [13] and WirelessHART [3] do not make use of it. Instead, these systems support concurrent transmissions using several 802.15.4 channels to reduce epoch length.

### 2.2 Capturing Worst-Case Link Reliability

There are different methods available to describe link reliability. Common methods are Expected Transmission Count (ETX) or Packet Reception Rate (PRR). Using PRR gives a worst-case link reliability by a value $P_{max}$ indicating that at least $P_{max}$ transmissions out of $n$ transmissions are successful. The problem with such a metric is that it does not capture the position of losses within the sequence of $n$ transmissions. For example, the schedule allowing for retransmissions shown in Fig. 1 is not valid if transmissions in two or more successive TDMA slots fail. $P_{max}$ might be large compared to $n$ indicating a good quality link. However, this might not be entirely true if losses appear in bursts. Unfortunately, this is exactly what can be observed in practice: losses cluster [7].

It has been shown that burst lengths [7] are a much better metric to capture worst-case link reliability for networks that have to support time-critical data delivery. We define worst-case link reliability using the two values $B_{max}$ and $B_{min}$: a link has no more than $B_{max}$ consecutive transmission errors and provides at least $B_{min}$ consecutive successful transmissions between two error bursts. It is possible to determine $B_{max}$ and $B_{min}$ before network deployment and to determine a schedule that can handle the *observed* worst-case [7, 13]. Figure 2 shows the schedule for the example topology for $B_{max} = 2$ and $B_{min} = 1$. Again, this schedule can only be used if $E \leq T$.

### 2.3 Evaluating Worst-Case Link Reliability

During a deployment a schedule based on $B_{max}$ and $B_{min}$ may become invalid as channel conditions change for the worse. Likewise, a schedule may become inefficient as channel conditions improve. It is therefore desirable to track the development of $B_{max}$ and $B_{min}$ over time in order to be able to adapt the deployed schedule if necessary. Alternatively, it might be possible to identify and remove the cause of a link quality degradation.

Nodes could generally use the transmission slots that are assigned to them to test $B_{max}$ and $B_{min}$ in every epoch. However, most nodes within the network

are not allocated sufficient transmission slots to determine an accurate reading of $B_{max}$ and $B_{min}$. For example, nodes $A$ and $C$ in the topology shown in Fig. 2 have only 3 consecutive transmission slots available which would not allow detection of a change from $B_{max} = 2$ and $B_{min} = 1$ to $B_{max} = 3$ and $B_{min} = 1$.

To ensure that all nodes can accurately measure $B_{max}$ and $B_{min}$ it is necessary to assign them a sufficient number of consecutive slots within the TDMA schedule. In most scenarios it is not possible to supply all nodes in this manner because the TDMA epoch $E$ would grow to exceed the time bound $T$ required by the application.

### 2.4 BurstProbe

We propose to use dedicated probe slots to evaluate $B_{max}$ and $B_{min}$ during network operations. The usage of a set of probe slots is called BurstProbe. Probe slots are located at the end of the epoch within the TDMA schedule and are shared among nodes. Nodes are assigned temporary ownership of the probe slots which they subsequently use to measure link burstiness. A data packet and potentially a corresponding acknowledgment is transmitted in each probe slot and the probe sender records the success pattern. The allocation of probe slots to nodes can either be determined automatically or by a user that decides to gather data on specific links. Figure 3 shows the schedule for the example topology including 4 probe slots at the end of the schedule.

The measurement of $B_{max}$ and $B_{min}$ is carried out at a different point in time than a data transmission occurs. In the example shown in Fig. 3 node $A$ transmits data in slots $s_1$, $s_2$ and $s_3$ at the beginning of an epoch, while link burstiness is measured in slots $s_{13}$ to $s_{14}$ at the end of an epoch. BurstProbe is only able to capture link burstiness if burst errors on a link are evenly distributed. However, our evaluation given in Sec. 4 shows that this is the case in real deployments and that BurstProbe is an effective measurement tool.

As probe slots are shared in the network, a node will not have access to them in every epoch (unless the node is assigned exclusive access). Hence, $B_{max}$ and $B_{min}$ is not tested in every TDMA epoch. However, link burstiness in practical deployments does not tend to change quickly but rather over many TDMA epochs. Thus, measurement of $B_{max}$ and $B_{min}$ in every other epoch is sufficient to obtain an accurate picture of link burstiness development over time. This is shown in the evaluation in Sec. 4 where we analyze real-world deployments using BurstProbe.

### 2.5 BurstProbe Effectiveness and Cost

The effectiveness of the BurstProbe mechanism is governed by the number of probe slots, the frequency with which probe sequences are executed and the nature of interference. Generally, the BurstProbe mechanism is more likely to determine an accurate $B_{max}$ and $B_{min}$ if a large number of probe slots are used and probe sequences are executed frequently. Infrequent usage of BurstProbe is feasible if the interference patterns are present for long periods.

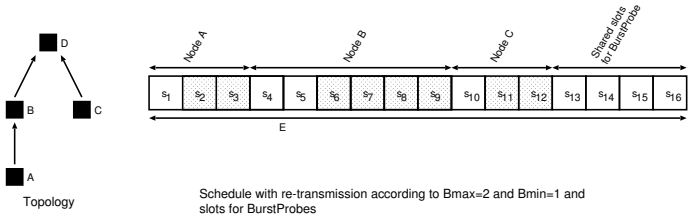



**Fig. 3.** Simple topology and possible schedules for lossy channels with $B_{max} = 2$ and $B_{min} = 1$ and 4 slots for BurstProbes.

The usage of BurstProbe causes additional energy costs. Firstly, a node must be active in additional slots to transmit and receive probe messages (Probing Cost). Secondly, the handling of the collected measurement data is energy costly as measurement data is stored locally (Storage Cost). Thus, the usage of BurstProbe reduces nodes lifetime.

Our experiments (see Sec. 4) show that probing costs are significant. The duty cycle of a node further away from the sink doubles in realistic settings. However, it has to be noted that overall duty cycles are still very low. Storage costs in all cases are generally negligible.

### 2.6 BurstProbe Limitations and Optimisations

The outlined BurstProbe mechanism is only able to measure interference on a link properly if the interference occurs during the time the probes are executed. Strict periodic interference which falls in the transmission slots of a node but never within the probe slots cannot be detected. This limitation can be resolved by introducing a dynamic TDMA schedule where the exact schedule is calculated by all nodes at the start of an epoch. This would allow us to move probe slots to the location of the transmission slots. Essentially, the number of available transmission slots for a specific node would be temporarily increased to ensure that data transmission and probing occurs at the same point in time. Although such a mechanism could be implemented, the resulting system would be very complex. However, as our experiments in Sec. 4 show it is not necessary to implement BurstProbe in such a way; in typical deployments probing slots at the end of the schedule produce useful measurement results.

In addition, Burstprobe is designed for use in scenarios which have static or slow changing topologies. It does not provide any procedure to automatically distribute new schedules to nodes in a deployment (this type of procedure would need to be defined specifically for the TDMA system employed if required). However, it can be used to gather the data necessary to devise new schedules when needed.

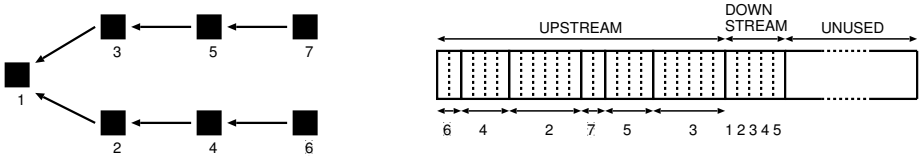



**Fig. 4.** 7 node example topology. $S_E = 100$ slots are used within the GinMAC epoch.

## 3 BurstProbe Implementation

We implemented BurstProbe for GinMAC [13], a state of the art solution for time-critical data delivery in wireless sensor networks.

### 3.1 GinMAC

GinMAC is a wireless sensor networking MAC protocol that has been specifically designed to support time-critical application scenarios. Currently, GinMAC is deployed at an oil refinery in Sines, Portugal [12] where it is used to connect a number of sensors and actuators used to monitor and control product flow and processing. Nodes are grouped in small networks running GinMAC at different frequencies (called cells). Cells are interconnected using a wired backbone infrastructure. The use of cells ensures that GinMAC networks are relatively small, this is necessary to obtain short epochs $E$ and tight delay bounds. GinMAC includes three main features of particular relevance to this task: Offline Dimensioning, Exclusive TDMA and Delay Conform Reliability Control. A network dimensioning process is carried out before the network is deployed. The input for the dimensioning process are network and application characteristics that are assumed to be known before deployment. The output of the dimensioning process is a TDMA schedule with epoch length $E$ that each node has to implement.

The GinMAC TDMA epoch consists of three types of slots: *basic slots*, *additional slots* and *unused slots*. First, the epoch contains a number of *basic slots* which are selected such that within frame length $E$ each sensor can forward $i$ messages to the sink and the sink can transmit $k$ messages to each actuator that might be present. Second, the GinMAC epoch uses *additional slots* to improve transmission reliability by providing capacity for retransmissions. Finally, the epoch may contain *unused slots* which are purely used to improve the duty cycle of nodes. The above types of slots within the GinMAC epoch must be designed such that the delay, reliability and energy requirements are met. However, it may not always be possible to find a schedule that simultaneously fulfils all requirements. The epoch $E$ might be too long and thus application delay targets cannot be met. If that is the case, some dimensioning assumptions must be relaxed.

To facilitate the description of how the GinMAC protocol operates we provide here an example of how a simple wireless sensor networking topology is supported using GinMAC. Consider the deployment of the wireless sensor network topology with 7 nodes (including sink) depicted in Fig. 4. At deployment

time the tree topology shown is determined to be feasible. All links are evaluated and $B_{max} = 1$ and $B_{min} = 1$ as worst-case on all links is determined. Next, the delay requirement is obtained from the application; for this example we assume that all nodes must be able to transmit one message within $T = 1s$ to the sink node. Using a slot length of $10ms$ an epoch with $S_E = 100$ slots is feasible. Next the number of basic and additional slots can be computed. Slots are allocated starting from the leaf nodes. Node 6 is assigned the first 2 slots in the epoch; one for data and one for a potential retransmission. Node 4 is assigned the next 4 slots to accommodate transmission of data from node 6 and 4 including potential retransmissions. To accommodate transmissions and potential retransmissions a total of 24 slots are allocated for basic and additional slots to accommodate upstream traffic. Finally, 5 slots are necessary to allow one broadcast message to be forwarded from the sink to each node within an epoch. The downstream slots are used within GinMAC to perform time synchronisation of all nodes with the sink. Tight time synchronisation is necessary to implement an effective TDMA schedule. In the example, 29 out of 100 available slots are used for data transmissions. The remaining 71 slots are unused and can be used to optimise the nodes duty cycle and to grant application processes execution time. The resulting TDMA schedule is shown in Fig. 4.

The unused slots in GinMAC represent an ideal opportunity for introducing probing functionality into its transmission schedule without the risk of disrupting its primary data transmissions.

### 3.2 BurstProbes in GinMAC

Within the GinMAC protocol the most appropriate place to insert probes is the area of unused slots at the end of the active slots. We extended GinMAC such that a variable number of probe slots can be added at the end of the active slots. When BurstProbe is executed the result of the probe sequence is recorded in the node's flash memory.

The inclusion of probe slots within the schedule is not problematical. However, the recording of the BurstProbe measurement results on flash memory introduces a number of challenges. Nodes have a finite amount of flash memory which will be filled over time. At some point it is necessary to clear the used space to enable further writing. With flash memory entire sectors must be cleared first before they can subsequently be reused which requires relatively long sector clearance operations to be performed. The T-mote Sky, the mote used to execute the GinMAC implementation, has a flash capacity of 1MByte which is split into 16 sectors of size of 64KB, a sector erase cycle typically takes $1S$. Assuming only 50% of the flash is used to record probe data, 52,500 probe patterns could be written before such erasing cycles would be needed. In the configuration above this would occur after only approximately 102 hours of use and would occur many times over the life time of the network whilst still supporting data transmission. Therefore, it is essential to execute the blocking clearance operations such that the time-critical TDMA schedule is not disturbed.

## 4 Debugging GinMAC with BurstProbes

In this section we show that BurstProbe is a very useful tool to accurately monitor changes in link burstiness over time. These observations can be used to either refine the TDMA schedule or to identify and remove the interference source. We also measure the energy consumption of the BurstProbe mechanism.

### 4.1 Debugging Scenarios

The GinMAC protocol is designed for industrial process automation and control applications. For example, it is used to monitor production processes in an oil refinery [13]. In such a setting a number of typical events can be observed which have an impact on link quality within a deployed network. Typical events are:

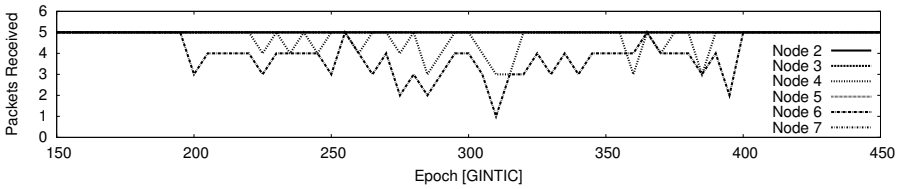
1. Obstacle: An obstacle obstructs (temporarily) communication and link quality degrades. For example, a truck of a maintenance crew is parked temporarily within a communication link or a new production unit is installed obstructing communication.
2. Interference: Other wireless communication devices or machinery interferes (temporarily) with transmissions on links. For example, other networks or machinery such as pumps may interfere with communication.

The aim of BurstProbe is to identify and quantify these events such that a new TDMA schedule can be computed. In particular, we are interested in adjusting the number of retransmission slots as discussed in the previous section. Alternatively, if no valid schedule can be computed (due to severe link degradation) the aim of BurstProbe is to then provide debugging information to help identify and remove the source of link quality degradation. We evaluate the capability of BurstProbe to deal with both of these events.

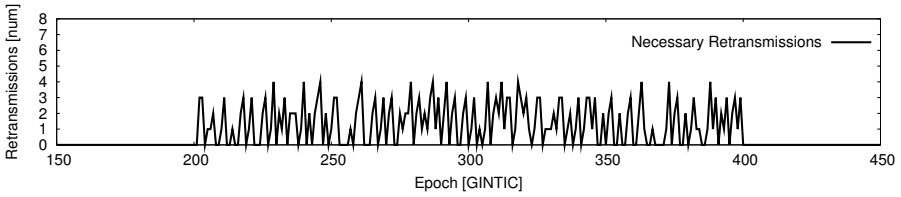
For evaluation we use a simple network consisting of 7 nodes as shown in Fig. 4. For the experiments a schedule with $S_E = 100$ slots of length $10ms$ is used which results in an epoch time of $1s$. A variable number of transmission slots and probe slots are used in each of the executed experiments. Probe slots are assigned to nodes in a round robin fashion; each node carries out a probe sequence every 7 epochs. Probe results are recorded in the node's flash memory and can be analyzed offline. For the purpose of evaluating the BurstProbe mechanism each node also records the number of retransmissions required in each epoch.

### 4.2 Interference on a Single Link

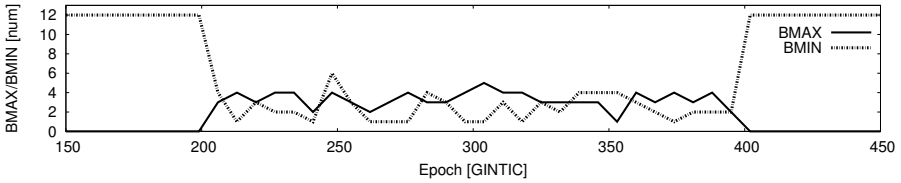
In the first experiment we configure the network for $B_{max} = 1$ and $B_{min} = 1$. As shown in Fig. 4 we need 29 transmission slots within the epoch of $S_E = 100$ slots. Initially, we use $S_P = 15$ slots for probing and the remaining 56 slots are unused. Each node transmits one data message within every epoch. In our experiments we run data transmissions for $t = 600s$. From $t = 200s$ to $t = 400s$ transmissions between node 4 and node 2 are disturbed. The disturbance is




a) Message reception over time at the sink node.




b) Retransmissions over time on link between node 4 and 2.



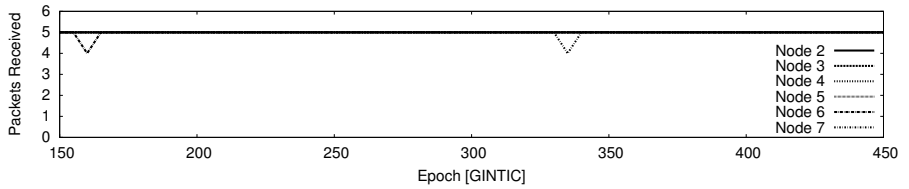

c) Recorded $B_{min}$ and $B_{max}$ values over time on link between node 4 and 2.

**Fig. 5.** Message receptions over time at the sink; re-transmissions over time at link $4-2$; BurstProbe result for link $4-2$.

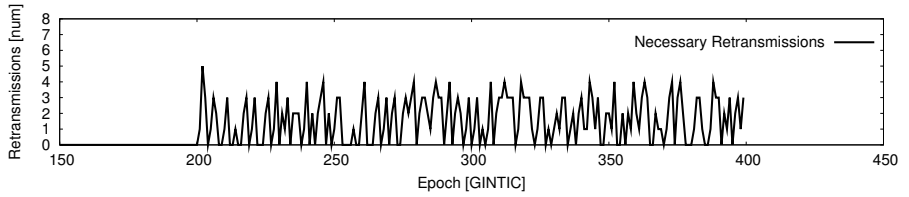
created by a purpose built interferer node that induces bursts of random size. Bursts have a worst-case characteristic of $B_{max} = 5$ and $B_{min} = 1$. Using an interferer node ensures that a ground truth can be established. The resulting link characteristic emulates a situation in which an interferer such as electric machinery or an obstacle would cause temporary link quality degradation.

Figure 5 a) shows the message reception over time as recorded at the sink node. Every 5 epochs the number of messages received per node over the 5 epoch time period is plotted. At first all messages generated by all nodes are received at the sink. When the interference starts messages generated by node 4 and node 6 are lost. After the interference stops message losses return again to zero. By observing message arrival at the sink it is only possible to determine that a network problem was present from $t = 200s$ to $t = 400s$. However, it is not possible to infer from observations at the sink the location of the problem and how it could be cured. For this, better means of network debugging are necessary.

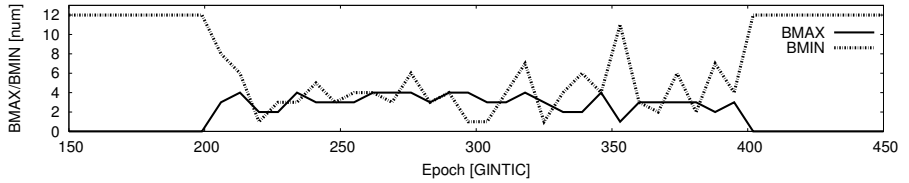
To gain more insight in the nature of the network problem we look at the necessary retransmission on links in the network. As messages have been lost some links must have used retransmission slots. The usage of retransmissions over time on the link between node 4 and 2 is shown in Fig. 5 b). Node 4 has 4 transmission slots available to transmit 2 messages every epoch to node 2. If




a) Message reception over time at the sink node.




b) Retransmissions over time on link between node 4 and 2.




c) Recorded $B_{min}$ and $B_{max}$ values over time on link between node 4 and 2.

**Fig. 6.** Message reception over time at the sink; re-transmissions over time at link $4-2$; BurstProbe result for link $4-2$ using the re-provisioned epoch.

more than 2 retransmissions are recorded within an epoch messages must have been lost. Node 6 is affected most by the link degradation as node 4 aims to deliver its own message first before forwarding messages of child nodes. The recordings of retransmissions on all other links in the network do not reveal retransmissions which shows that something must have interfered exclusively on the link between node 4 and node 2 at $200s < t < 400s$. However, even such a detailed recording of retransmissions over time does not help in determining how the TDMA schedule should be configured to deal with the problematic link. From the logs we can infer that the selected values for $B_{max}$ and $B_{min}$ on link $4-2$ are not correct. We cannot, however, infer the true value of $B_{max}$ and $B_{min}$ with this approach.

To obtain the true value of $B_{max}$ and $B_{min}$ we use BurstProbe. Figure 5 c) shows the recording of BurstProbe results obtained every 7 epochs on link $4-2$. Between epoch 200 and 250 $B_{max}$ increases to a value of 4 while $B_{min}$ drops to 2. The worst-case over the affected period is a $B_{max} = 5$ and $B_{min} = 1$. This measurement reflects the worst-case interference induced by the interfering node. With this data it is now possible to decide on corrective measures. A first option is to correct the schedule to include slots for the appropriate amount of potential retransmissions on link $4-2$. A second option would be to exclude the link from

**Table 1.** Radio Duty Cycle with and without BurtsProbes.

| Node | Standard | Probes | Increase |
|---|---|---|---|
| 2 | 4.11% | 5.34% | 1.23% |
| 4 | 3.02% | 4.77% | 1.75% |
| 6 | 0.99% | 2.87% | 1.88% |

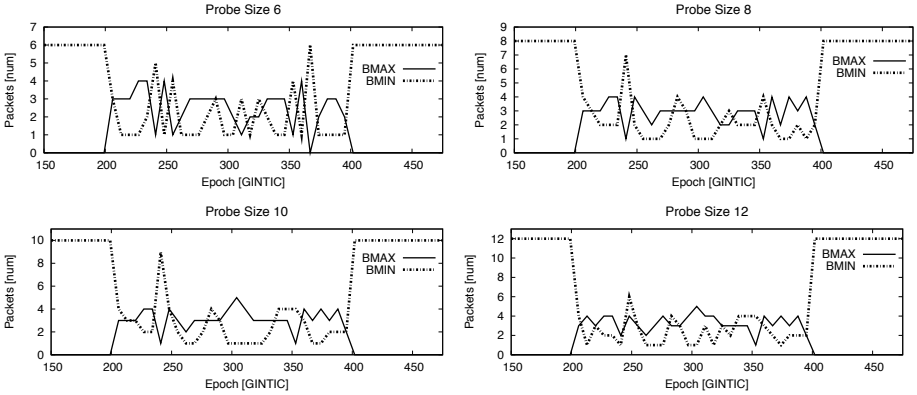



**Fig. 7.** $B_{min}$ and $B_{max}$ measurement with $S_P = \{6, 8, 10, 12\}$ probe slots.

the topology if it is considered to be of a too poor quality. A third option would be to investigate the deployment to see if a potential interferer or an obstacle can be removed. For the purpose of this experiment, we decide to correct the TDMA schedule for link $4-2$ assuming a $B_{max} = 4$ and $B_{min} = 1$. As only one measurement showed a $B_{max} = 5$ we do not include it in the corrections.

We repeated our experiment after applying the corrections. Link $4-2$ now provides 10 transmission slots to handle $B_{max} = 4$ and $B_{min} = 1$. Fig. 6 a) shows that almost all messages are delivered to the sink. Figure 6 b) depicts that as expected 3 or 4 retransmissions are required. The probe results give the same indication on $B_{max}$ and $B_{min}$. As we dimension for $B_{max} = 4$ and $B_{min} = 1$ the rare worst-case of $B_{max} = 5$ and $B_{min} = 1$ is not captured and some losses still occur. In summary, BurstProbe enables us to accurately determine a schedule that is able to handle the link quality degradation in the period $200s < t < 400s$.

We also measured the energy consumption of BurstProbe. First, we run the experiment with BurstProbe disabled and measure the nodes' transceiver usage time and flash usage time. Thereafter, we repeat the experiment with the BurstProbe mechanism. Our results show that the flash usage time for all nodes increases by approximately $20ms$ over the $600s$ experiment. This overhead is very small given that it is the equivalent of transmitting 4 additional packets over the 10 minute experiment. The changes in the transceiver duty cycle are shown in Tab 1. Based on the additional slots we expected an increase of 1.9% but the increase was slightly less due to traffic fluctuations caused by interference.

It is possible to decrease the energy consumption for the BurstProbe mechanism by reducing the number of probe slots. To investigate the effect of the number of probe slots on the accuracy of $B_{min}$ and $B_{max}$ we ran the experiment with $S_P = \{6, 8, 10, 12\}$ probe slots. The results are shown in Fig. 7. Only for $S_P = 6$ BurstProbe cannot identify the worst-case of $B_{max} = 5$ and $B_{min} = 1$ . Thus, a probe number of $S_P = 8$ is sufficient for our application scenario. With $S_p = 8$, the energy cost for BurstProbe for node 4 decreases from 4.77% with $S_P = 15$ to 3.87%.

### 4.3 Network Wide Interference

In the second experiment we configure the network with the same initial configuration as in the first experiment in Sec. 4.2. The network has 7 nodes, using $B_{max} = 1$ and $B_{min} = 1$, with an epoch of size $S_E = 100$ slots and $S_P = 15$ probe slots. The experiment was deployed in the shape of an L, centred at the sink with each of the two branches running 90 degrees away from one another. As shown in Fig. 4 the first branch consisted of nodes 2, 4, 6 whilst the second had nodes 3, 5, 7. The network is used for $t = 600s$ to transmit data from each node to the sink at a rate of one packet per epoch per node. From $t = 200s$ to $t = 400s$ a Wi-Fi network occupying the same frequency as the network is enabled which generates interference. The Wi-Fi access point is located in the vicinity of node 3.

Figure 8 shows both the BurstProbe transmission results and the number of retransmissions recorded by each node. The figure is divided into two columns and three rows of smaller plots. The first column has the recorded values of branch one whilst the second has the values of branch two. With each row, the distance from the sink increases. The figure shows significant interference on links $2-1$, $3-1$, $5-3$ and $7-5$ where on a number of occasions all 15 probes were recorded as lost. At these points in time it is impossible to accurately calculate the values of $B_{max}$ and $B_{min}$. The worst-case value for $B_{max}$ and $B_{min}$ where their value could be calculated was $B_{max} = 14$ and $B_{min} = 1$. This significant interference observed by the probe measurements was also seen in the high number of retransmissions recorded at each of the above links and high packet loss rate observed at the sink.

Due to the significance of the interference experienced, resolving the issues in a similar fashion to the first experiment is difficult. The worst-case values of $B_{max}$ and $B_{min}$ were not observed in the experiment as 15 probes proved insufficient. Furthermore, using the worst-case observable value of $B_{max} = 14$ and $B_{min} = 1$ to re-provision the network would require 197 transmission slots. This is more slots than are available within the epoch. Although the epoch size could be increased, as the epoch size increases so does the message delivery delay. Here the required epoch size would increase the communications delay beyond the value acceptable to the application. Therefore, simply re-provisioning the network is not a viable solution under the interference observed.

The second option to addressing interference issues, discussed in Sec. 4.2, is to remove problem links. The interference is widespread and occurs on the majority

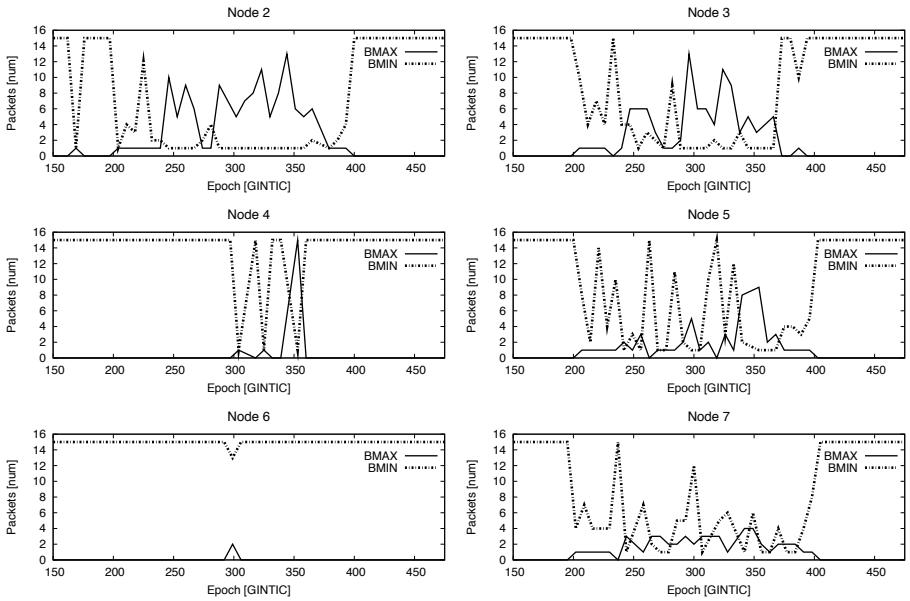



**Fig. 8.** Network wide interference caused by Wi-Fi.

of links, simply removing links would not lead to a suitable solution. Therefore the only solution here is to identify and remove the source of the interference.

Figure 8 shows that the interference was recorded as being more prominent on links $2-1$, $3-1$, $5-3$ and $7-5$. This information could be used in a real deployment to identify the location of the interference source. Examining Fig. 8 we can deduce that the interference source must be in the vicinity of the sink and the first branch as the interference in the second branch reduces with distance from the sink. This would provide valuable information in pinpointing the location of the interference to eradicate it. These results confirm the actual location of the Wi-Fi interference source as in the vicinity of node 3.

## 5 Related Work

To date WSN research has produced a number of solutions aimed at addressing timely data delivery in wireless sensor networks [7, 3, 4, 13]. All of these solutions require precise knowledge of available link quality in order to select transmission schedules. Thus, the BurstProbe mechanism outlined in this paper may be applied to any of these solutions. With regards to the BurstProbe approach and the task of debugging wireless sensor networks in general, earlier techniques used for analysing performance problems relied on the data sink retrieving live debug data from each node in the network [10]. Other techniques used additional nodes to monitor radio traffic and problems [1] [11]. The BurstProbe approach is based

on the concept of inserting dedicated probes into unused transmission slots and recording the probe result patterns locally. Other examples where performance data is stored locally on motes include [8] where an approach for diagnosing performance anomalies is presented that highlights the potential benefits of using local flash storage of system data, and Envirolog [6] that is designed to allow the user to produce an exact replay of recorded events and conditions to aide performance evaluation. Also related are [5] and [9] which embed performance related data in application messages to employ a passive approach to anomaly detection. PD2 [2] takes an alternative approach by focusing on data flows that individual applications generate, relating poor application performance to loss and latencies of data flows. This allows performance monitoring and debugging information to only be enabled on the nodes that data flows are known to traverse, thus reducing the overall overheads imposed. However, whilst all of these proposed approaches offer different techniques and models for recording and in some cases disseminating performance information, BurstProbe provides novelty by provisioning specific transmission slots to insert dedicated probing that can help determine more accurate information about loss and retransmissions that are occurring throughout a wireless sensor network deployment.

## 6 Conclusion

In this paper we presented BurstProbe, a mechanism useful to debug time-critical WSNs. As shown, BurstProbe is a useful tool for measuring changes in link burstiness over time within a running network. The BurstProbe measurements can be used to correct TDMA schedules or to locate sources of interference. A key characteristic of the BurstProbe mechanism is that it can be included in WSN systems such as GinMAC or WirelessHART without interfering with the time-critical data delivery. The cost of the BurstProbe mechanism in terms of energy cannot be neglected; in the described experiments node duty cycles increase in the order of 2%. However, we believe that such relatively modest energy investment is necessary in order to be able to debug WSN for time-critical data delivery. In our prototype deployment at an oil refinery in Sines, Portugal the resulting node lifetime of a few months is acceptable as default system maintenance is carried out frequently. In the paper we discussed the basic functionality of BurstProbe but many optimisations and refinements are not explored. Firstly, it is necessary to investigate scheduling mechanisms for probe slots. It would be useful to schedule more probes on links that currently experience problems while reducing probe frequency on good links. Secondly, it would be useful to implement a mechanism to fetch recorded burst probe data remotely rather than collecting nodes and extracting the data from the flash manually for analysis. A mechanism as described in [9] might be a good starting point for such extension.

## Acknowledgment

This work has been partially supported by the EC under the FP7 contract ICT-224282 (GINSENG) and the FP7 contract ICT-224053 (CONET).

## References

1. B. Chen, G. Peterson, G. Mainland, and M. Welsh. Livenet: Using passive monitoring to reconstruct sensor network dynamics. In *Proceedings of the 4th IEEE international conference on Distributed Computing in Sensor Systems*, DCOSS '08, 2008.
2. Z. Chen and K. G. Shin. Post-deployment performance debugging in wireless sensor networks. In *Proceedings of the 2009 30th IEEE Real-Time Systems Symposium*, RTSS '09, 2009.
3. HART Communication Foundation. Wirelesshart data sheet. http://www.hartcomm.org/, 2010.
4. International Society of Automation (ISA). Isa100 wireless systems for automation. http://www.isa.org/, 2010.
5. K. Liu, M. Li, Y. Liu, M. Li, Z. Guo, and F. Hong. Passive diagnosis for wireless sensor networks. In *Proceedings of the 6th ACM conference on Embedded network sensor systems*, SenSys '08, 2008.
6. L. Luo, T. He, G. Zhou, L. Gu, T. F. Abdelzaher, and J. A. Stankovic. Achieving repeatability of asynchronous events in wireless sensor networks with envirolog. In *Proceedings of the IEEE Conference on Computer Communications*, INFOCOM '06, 2006.
7. S. Munir, S. Lin, E. Hoque, S. M. Shahriar Nirjon, , J. A. Stankovic, and K. Whitehouse. Addressing burstiness for reliable communication and latency bound generation in wireless sensor networks. In *Proceedings of the 9th ACM/IEEE International Conference on Information Processing in Sensor Networks*, IPSN '10, 2010.
8. T. O'Donovan, N. Tsiftes, Z. He, T. Voigt, and C. J. Sreenan. Detailed diagnosis of performance anomalies in sensornets. In *Proceedings of the ACM Workshop on Hot Topics in Embedded Networked Sensors*, HOTEMNETS '10, 2010.
9. V. Pejovic and C. J. Sreenan. Perdb: Performance debugging for wireless sensor networks. In *Proceedings of the European Conference on Wireless Sensor Networks, Poster/Demo session*, EWSN '09, 2009.
10. N. Ramanathan, K. Chang, R. Kapur, L. Girod, E. Kohler, and D. Estrin. Sympathy for the sensor network debugger. In *Proceedings of the 3rd international conference on Embedded networked sensor systems*, SenSys '05, 2005.
11. M. Ringwald, K. Römer, and A. Vitaletti. Passive inspection of sensor networks. In *Proceedings of the 3rd IEEE International Conference on Distributed Computing in Sensor Systems*, DCOSS '07, 2007.
12. C. J. Sreenan, J. SaSilva, L. Wolf, R. Eiras, T. Voigt, U. Roedig, V. Vassiliou, and G. Hackenbroich. Performance control in wireless sensor networks: the ginseng project - [Global communications news letter]. *Communications Magazine*, 47(8), August 2009.
13. P. Suriyachai, J. Brown, and U. Roedig. Time-critical data delivery in wireless sensor networks. In *Proceedings of the 6th IEEE International Conference Distributed Computing in Sensor Systems*, DCOSS '10, 2010.