Addressing Burstiness for Reliable Communication and Latency Bound Generation in Wireless Sensor Networks

Sirajum Munir, Shan Lin, Enamul Hoque, S. M. Shahriar Nirjon, John A. Stankovic, and Kamin Whitehouse
Department of Computer Science, University of Virginia, Charlottesville, VA, USA
{munir, shanlin, enamulhoque1, nirjon, stankovic, whitehouse}@cs.virginia.edu

ABSTRACT

As wireless sensor networks mature, they are increasingly being used in real-time applications. Many of these applications require reliable transmission within latency bounds. Achieving this goal is very difficult because of link burstiness and interference. Based on significant empirical evidence of 21 days and over 3,600,000 packets transmission per link, we propose a scheduling algorithm that produces latency bounds of the real-time periodic streams and accounts for both link bursts and interference. The solution is achieved through the definition of a new metric $Bmax$ that characterizes links by their maximum burst length, and by choosing a novel least-burst-route that minimizes the sum of worst case burst lengths over all links in the route. A testbed evaluation consisting of 48 nodes spread across a floor of a building shows that we obtain 100% reliable packet delivery within derived latency bounds. We also demonstrate how performance deteriorates and discuss its implications for wireless networks with insufficient high quality links.

Categories and Subject Descriptors

C.2.1 [**Computer-Communication Networks**]: Wireless Communication; C.2.2 [**Computer-Communication Networks**]: Network Protocols; C.3 [**Special-Purpose and Application-Based Systems**]: Real-time and Embedded Systems

General Terms

Algorithms, Measurement, Performance, Reliability

Keywords

Link Burstiness, Link Interference, Latency Bound, Reliable Transmission, Real-time Applications.

IPSN'10, April 12–16, 2010, Stockholm, Sweden.

1. INTRODUCTION

Wireless sensor networks (WSN) is an exciting enabling technology that allows monitoring, communication and control for an ever increasing set of applications. Many such applications will use open and heterogeneous wireless networks with sensors and actuators. In most of the cases, these sensors and actuators will form a multi-hop ad hoc network where streams of data flow from one hop to another. However, the wireless links are highly non-deterministic because of link burstiness and interference. Link burstiness is a physical property which means that transmissions on a wireless link do not have independent probability of failure; instead they have periods of continuous message loss, i.e. they fail in a burst. Link interference is another physical property of the communication environment which causes packet transmission between different links to interfere with each other which results in packet loss. Due to these types of non-determinism, it is difficult to offer reliability and latency bounds for packet delivery over wireless networks.

However, reliability and end-to-end latency bounds are important for many applications because of real-time requirements. For example, consider applications with real-time constraints like target tracking [10], and industrial monitoring [8]. In such applications, streams need to reach the destination within a specific time. Let us consider the following motivating example. Many Industrial control plants, e.g., chemical process control, require distributed sensor networks to monitor and control the plant. Communications must be reliable and delivered on-time. Traditionally, such applications use statically scheduled wired networks. However, the cost of wiring and re-wiring as the plant expands or shrinks is expensive. Re-setting sensors or actuators is also made difficult by the wiring. If a WSN can provide reliable real-time communication and on time delivery, the cost benefits and flexibility would be large.

Achieving reliable WSN communication is very challenging. To accomplish it, at first we characterize the physical properties like interference and burstiness of the particular network. Then we can schedule packet transmission in a way that overcomes the difficulties offered by these properties of the communication environment. It is obvious that we cannot allocate only a single transmission time slot for a stream on each link, especially if we are dealing with bursty links. Because, if the transmission fails at that time slot due to a link burst, the node will need some additional time slots to transmit its packet. So, to provide the end-to-end latency bounds, we have to allocate more than one time slot per link for a stream. The number of time slots we need to al-

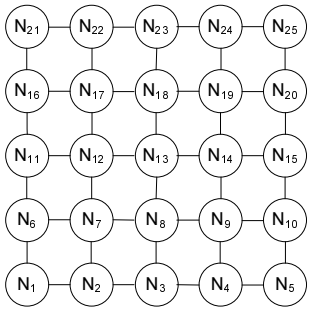

Figure 1: An Example Topology

locate depends on the burstiness of the link. We don't want to allocate more time slots than we need, otherwise we will increase the latency bound. In addition, because of interference, other streams cannot be scheduled in nearby links during that entire multiple slot allocation time, which may increase the overall latency bound of all streams. So, achieving reliable communication and minimizing latency bound by scheduling is therefore a challenging goal.

The specific problem that we are addressing assumes that we are given a network topology and a set of periodic streams. We calculate an upper bound of the latency for each of the streams by taking into account both link burstiness and interference. Current state of the art approaches can not bound latency, because most of the approaches use PRR based time slot allocation that fail to account for link bursts. Our 21 day long empirical study shows that over 23% links having PRR as high as 0.99 lose more than 50 packets in a row, some lose even over 1000 packets in a row. So, the PRR based approach can not produce a bound on latency. We need to carefully design some other metrics to characterize the burstiness of the links that will allow us to allocate sufficient number of time slots to produce a latency bound.

This paper has three research contributions. First, based on 21 days of empirical study over a 802.15.4 network we define a new metric *max burst length (Bmax)* that allows us to classify links and allocate sufficient number of time slots to produce a latency bound of the streams. Empirical evidence shows that stationarity of link quality can be better characterized by $Bmax$ than PRR. Second, we design a static network-wide stream scheduling algorithm that uses a novel least-burst-route to produce latency bounds of the streams by taking into account link bursts and interference. Finally, by using testbed evaluation we show that we can acutally bound the latency by achieving 100% packet delivery ratio within the derived latency bounds. We also investigate how the performance degrades when we do not have sufficient high quality links in the network.

An implication of these contributions is that if each of the streams has period greater than or equal to the latency bound that we provide, then our scheduling algorithm allows reliable communication subject to our burstiness and interference assumptions. If the burstiness characterization used for creating the schedules is violated during the course of execution, then a deadline might still be missed, but this is rare because the link characterization is performed under realistic operating conditions and an adaptive solution will reduce such scenarios subsequently. Note that our approach does not minimize latency to maximize throughput.

Stream	Period	ST	Source	Dest.	Route
S_1	20	1	N_1	N_4	N_1, N_2, N_3, N_4
S_2	20	1	N_2	N_5	N_2, N_3, N_4, N_5
S_3	20	1	N_7	N_9	N_7, N_8, N_9
S_4	10	1	N_{17}	N_{19}	N_{17}, N_{18},N_{19}

Table 1: An Example Set of 4 Streams

Instead, our average delivery latency is higher than most other techniques. However, we do offer a reliable communication and latency bound, which makes it easier to engineer predictable systems. We verify this claim by evaluating our approach on a 48-node wireless testbed with 10 simultaneous and periodic packet streams that shows our scheme has a 100% on-time delivery ratio.

The rest of this paper is structured as follows. Section 2 formally defines the problem. Section 3 describes related work on scheduling streams with real-time constraints in wireless sensor networks. Section 4 describes our model parameters, assumptions and link classification based on an empirical study. Section 5 describes the scheduling algorithm that computes the latency bound. Section 6 describes the experimental setup and the results of the experiments. We conclude in Section 7.

2. PROBLEM DEFINITION

The problem that we are addressing is formally stated as follows: given a number of periodic streams and a network topology, calculate an upper bound on the latency of all the streams so that all the packets of all the streams reach their destinations within their respective latency bounds. We assume stationary nodes and a fixed topology.

We define a Stream Set SS that contains n periodic streams, where a periodic stream S_i has a source node SRC_i, a destination node $DEST_i$, a route RT_i, a starting time ST_i and a period P_i, where $i = 1, 2, 3, \cdots, n$ and a stream is represented as a 6-tuple (S_i,SRC_i, $DEST_i$, RT_i, ST_i, P_i). After the starting time is elapsed, at every period, the source node has a packet that has to be transmitted to the destination node by following that route.

Network topology is specified as a set of nodes N_1, N_2, N_3, $\cdots$, N_k along with their connectivity matrix and interference matrix. For each pair of connected nodes N_i and N_j, we denote the intermediate link as $L(i, j)$ that has burstiness parameters $Bmax$ and $B'min$ which are estimated based on empirical data. We will define these parameters formally in Section 4.

For example, consider the network topology shown in Figure 1. If two nodes are within their radio range, they are connected by an edge. Suppose that we are given 4 streams to calculate their latency bounds. The details of the streams are shown in Table 1. The route means stream S_1 is going from node N_1 to N_4 using route $N_1 \rightarrow N_2 \rightarrow N_3 \rightarrow N_4$. So, we have $SS = \{(S_1, N_1, N_4, RT_1, 1, 20), (S_2, N_2, N_5, RT_2, 1, 20), (S_3, N_7, N_9, RT_3, 1, 20), (S_4, N_{17}, N_{19}, RT_4, 1, 10)\}$ where $RT_1 = \{N_1, N_2, N_3, N_4\}$, $RT_2 = \{N_2, N_3, N_4, N_5\}$, $RT_3 = \{N_7, N_8, N_9\}$, $RT_4 = \{N_{17}, N_{18}, N_{19}\}$.

Given this input set, our algorithm schedules all the streams and outputs latency bound LB_i for each stream S_i. We claim that every packet of stream S_i will reach its destination no later than $ST_i + LB_i$ if packet transmission is performed according to our schedule.

3. RELATED WORK

Recent studies have shown that most of the wireless links are bursty. Srinivasan et al. [22] have studied 802.15.4 and 802.11b networks and presented a new metric β to measure link burstiness. He has shown that we can avoid link burstiness by having an interpacket delay of 500 ms. But our 21 days of study on a 802.15.4 network shows that we can do even better for some links i.e. for some links we don't have to wait that much of time to avoid packet loss due to link bursts.

Several research works have already been done to characterize wireless links. Statistical properties of low power wireless links have been analyzed in [4]. It shows that link failures have temporal characteristics. Characterization of signal strength properties is demostrated in [19] by using Monopole Antennas. It shows that antenna orientation effects are the dominant factor of the signal strength sensitivity. [16] presents a metric called competence that characterizes link for a longer time frame. It improves end-to-end delivery ratio and reduces energy consumption. E^2WFQ [20] offers fair packet scheduling. It is an energy saving version of Weighted Fair Queueing (WFQ) algorithm. ATPC [15] employs feedback-based dynamic transmission power control to adjust the quality of radio communicaiton. It also achieves more energy saving. ART [9] controls topology to reduce power consumption and channel contention. It can adapt to the variations in link quality and contention. [23] exploits spatial diversity in industrial wireless networks based on relaying approach. [2] presents a model that accounts for the correlation that exists in shadow fading between links in multihop networks. [24] presents Radio Irregularity Model (RIM) to analyze the impact of radio irregularity on the communication performance and provides solution to deal with this issue. [12] offers algorithm admission control and two scheduling policies that are proved to be feasibility optimal for wireless network based applications with QoS requiements.

There are a number of scheduling algorithms for stream transmission. But they have many assumptions or shortcomings that make them unsuitable for scheduling periodic streams with real-time constraints in WSN.

Previous performance analysis [3] shows that DSR [13] outperforms other ID-based routing protocols in terms of successful packet delivery ratio, but it does not consider time constraints. RAP [18] uses a velocity monotonic scheduling algorithm that takes into account both time and distance constraints. It maximizes number of packets meeting their end-to-end deadlines, but reliability aspects of individual streams are not addressed. SPEED [11] maintains a desired delivery speed across the sensor network by a combination of feedback control and non-deterministic geographic forwarding. It is designed for soft real time applications and it is not concerned with the reliability issues of individual streams. Also, our approach is based on static scheduling, while these approaches are based on dynamic scheduling. So, neither of these approaches seem to be appropriate to be compared against our approach.

WirelessHART [1] is a wireless mesh networks communication protocol designed to meet the needs for process automation applications. It combines several features to provide 99.9% end-to-end reliability in all industrial environments. The features include channel hopping on a message-by-message basis, monitoring paths for degradation and automatic repair, finding alternative paths around obstructions. But it does not address packet loss due to link bursts. As a result a latency bound cannot be provided.

RI-EDF [6] is a MAC layer protocol that provides a real-time guarantee by utilizing the rules of EDF [17] to derive a network schedule. But, in this work the network has to be fully linked, that is, every node is within direct transmission range of every other node. This constraint makes this protocol unsuitable for many networks. [14] provides timeliness guarantees to multi-hop streams by explicitly avoiding collisions and scheduling messages based on per-hop timeliness constraints in real-time robotic sensor applications. None of the above mentioned algorithms actually considers link burstiness that can affect the packet transmission significantly.

4. EMPIRICIAL STUDY

To see how burstiness affect packet transmission, we run a 21 days long experiment on a 48 node 802.15.4 network and transmitted 3,600,000 packets over every link. From these experiments we confirmed that burstiness in wireless packet transmission is a ubiquitous phenomenon. We also observed that for links with similar link quality, their burstiness behavior can be different. As a result, the transmission latency of data streams depends on burstiness of links that the streams go through.

4.1 Model Parameters and Assumptions

To characterize link bursts we define five parameters: B, $Bmax$, B', $B'min$, and W for every link. The way these parameters are computed and defined is as follows: after transmitting 3,600,000 packets over every link, we have a long sequences of data trace of 0s and 1s per link where 1 at ith index of the sequence means that packet with ith sequence number was successfully transmitted and 0 at that place means it failed. As an example, consider a sample data trace 0110010011 of length 10. Now, we define W as a window for packet transmission having length $|W|$. For this particular example data trace, assume that we have $|W| = 3$. So, we have 8 different windows to consider, each of length 3, where the first window spans from index 1 to 3 having values 011, the second window spans from index 2 to 4 having values 110, the third window spans from index 3 to 5 having values 100 and so on. For a particular window, we define B as the number of time slots where packet transmission failed due to link bursts and B' as the number of time slots where packet transmission was successful. So, for this particular example, we have $B = 1$, $B' = 2$ for the first window, $B = 1$, $B' = 2$ for the second window, $B = 2$, $B' = 1$ for the third window and so on. Now, for a particular window of size $|W|$, we define $Bmax$ as the maximum value of B for all possible windows of size $|W|$ and $B'min$ as the minimum value of B' for all possible windows of size $|W|$. So, for this particular example, $Bmax = 2$ and $B'min = 1$ where $|W| = 3$.

So, the key idea is, we define $Bmax$ as the maximum number of time slots where packet transmission can fail due to a burst and $B'min$ as the minimum number of time slots that are available for packet transmission between two consecutive bursts. So in a window size of $Bmax + B'min$, we have at least $B'min$ time slots for successful packet transmission. Note that different links have different burst characteristics and to obtain a particular $B'min$ we need to consider differ-

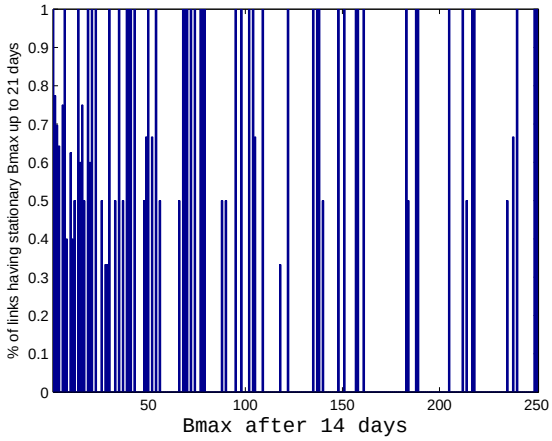

Figure 2: $Bmax$ as a Stationarity Metric

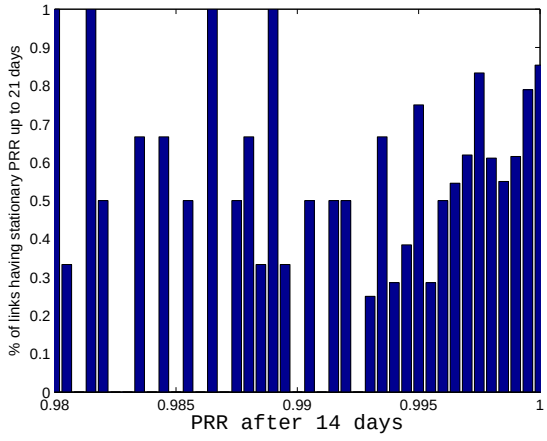

Figure 3: PRR as a Stationarity Metric

ent sized window for different links. *So, we are not defining a window for all the links, instead we are calculating the window size of a link for a particular $B'min$ by calculating first $Bmax$ and then computing $|W| = Bmax + B'min$.*

Algorithm 1 : ComputeBmax(D, $B'min$)

```
1: for i ← B'min + 1 to |D| do
2:    isSatisfied ← TRUE
3:    for j ← 1 to |D| − i do
4:       B' ← Σ_{k=j}^{j+i−1} D[k]
5:       if B' < B'min then
6:          isSatisfied ← FALSE
7:          break
8:       end if
9:    end for
10:   if isSatisfied = TRUE then
11:      Bmax ← i − B'min
12:      return Bmax
13:   end if
14: end for
15: return −1
```

Algorithm $ComputeBmax()$ computes $Bmax$ given a data trace D of 0s, 1s and $B'min$. It returns -1 if the data trace doesn't have a $Bmax$ that satisfies the condition specified by the definition of $Bmax$ amd $B'min$. The running time of the algorithm is $O(|D|^2)$ which is large for long data trace, although B' at line 4 can be computed in $O(1)$ time by using a dynamic programming based memoization approach. But if a link has a very high $Bmax$, it indicates that the link is prone to heavy bursts, and we avoid this link for real time applications. So, for practical purposes, we limit the maximum $Bmax$ to be $C = 1200$, and constrain the loop at line 1 to run from 1 to C. Then, the running time of the algorithm becomes $O(C|D|)$ which is linear with respect to the size of the data trace.

Now we describe the necessity and importance of using $Bmax$ for calculating the latency bound. Some links do have arbitrarily large $Bmax$ and such links are avoided in choosing routes in our methodology. So, we are only considering links having $Bmax \leq 1200$. Figure 4 demonstrates the $Bmax$ of the links (when $B'min = 1$) for different values of PRR. The figure shows that some links have large $Bmax$ although the corresponding PRR is as high as 0.99. Our experimental result shows that we have 32.72%, and 12.32% links having PRR as high as 0.99 and 0.999 suffer from a burst of having length ≥ 5. So, if we allocate time slots based on PRR, some packets may not get through due to a burst of "unexpected" length and the packet will miss its deadline. So, $Bmax$ is a better metric for allocating time slots to offer a latency bound compared to PRR as it is considering link bursts.

To demonstrate that links having low $Bmax$ tend to be more stationary and $Bmax$ can characterize non-stationary links better than PRR we present Figure 2 and Figure 3. We calculate $Bmax$ of the links after 14 days and 21 days and plot the percentage of the links having stationary $Bmax$ within this interval. Figure 2 shows that low $Bmax$ links tend to be more stationary. In contrast, we compute PRR of the links after 14 days and 21 days and plot the percentage of links having stationary PRR within this interval. We consider from links having PRR 0.98 and plot with an increment of 0.0005 in Figure 3 that shows that links having PRR as high as 0.99 don't preserve stationarity even over seven days.

Our model assumes that after network characterization, i.e., after we compute $Bmax$ and $B'min$ of every link, if we consider $Bmax + B'min$ time slots for packet transmission, we have at least $B'min$ time slots to transmit packet successfully. Although the assumption looks questionable, we have some strong arguments in favor of it. The assumption may not hold in a battlefield where link behavior may change drastically, but it seems to hold in a regular working environment like offices, universities, and industrial plants if we can characterize the links for a long period of time under all possible working environments. Obviously, this assumption will not hold for all the wireless links. We classifiy the links (in the next section) for which the assumption seems to be true. If the wireless links are not good enough to meet the assumption, we can move some nodes or add additional nodes in the network to create the "right" topology having

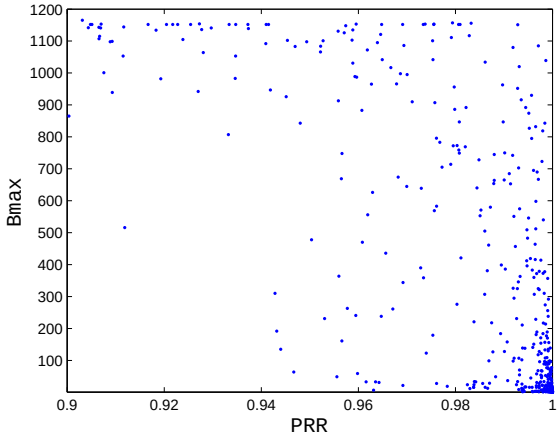

Figure 4: $Bmax$ vs. PRR

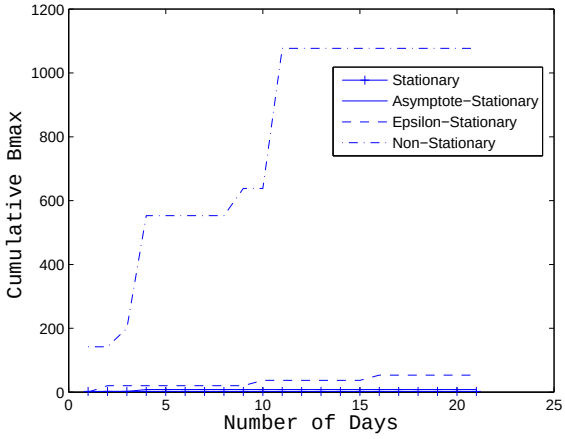

Figure 5: $Bmax$ vs. Days

good burst properties that satisfies the model assumption. Also, recent work [21] has explored that bursts in wireless link have scaling properties meaning that the bursts shows self-similarity or other coherent structure over many time scales without having long range dependence. The paper specifies an onset point of 640 ms where random variations stop affecting the wireless link and self-similarity starts to dominate. It clearly indicates the possibility of capturing the burstiness characteristics of the wireless links if we investigate the burstiness behavior of the wireless links over a long period of time.

To characterize link interference, we define an interference matrix, IM that represents the interference pattern for the network by specifying which links potentially interfere with others.

4.2 Link Classification

Based on the values of $Bmax$, we can classifiy the wireless links into 4 categories:

1. Stationary Links
2. Asymptote-Stationary Links
3. Epsilon-Stationary Links
4. Non-Stationary Links

We calculate $Bmax$ of the links for every day, and calculate the $cumulativeBmax$ which is the maximum $Bmax$ from the first day up to that day. Figure 5 shows how the $cumulativeBmax$ varies as the days proceed and allows us to distinguish between Stationary links and Asympote-Stationary links. We call the links having constant $cumulativeBmax$ from the first day as Stationary links. We call the links Asymptote-Stationary for which the $cumulativeBmax$ becomes constant after a few days (we set this threshold as 14 days). Epsilon Stationary and Non-Stationary links do not show stable $cumulativeBmax$ even after 14 days and they are largely affected by the physical environment. The probability of observing large bursts is very small for both cases, but in case of Epsilon Stationary links we have not observed any burst of having length > 1000 while in case of Non-Stationary links we have observed that within 21 days.

Figures 5 is not only important for link classification, it is also essential for calculating $Bmax$ that the scheduler needs to know to allocate time slots. We can only use Stationary and Asymptote-Stationary links for hard real time applications where the corresponding $Bmax$ will be computed from the asympotes of the curves from Figure 5. Epsilon-Stationary links can be used for soft real time applications with some probability of missing end-to-end deadline. For example, assume that an application requires 99% end-to-end packet delivery on time. The probability calculation may proceed from the distribution of B as follows: all we have to do is to find a minimum $Bmax$ that will allow us not to miss end-to-end deadline more than 1% of the time. If the route is n hop, then for every link L_i that the route goes through can have a $Bmax = b_i$ where the probability of observing a burst having length greater than b_i is less than or equal to $\sqrt[n]{0.01}$ i.e. $Prob[Bmax > b_i] \leq \sqrt[n]{0.01}$. Note that this approach allows us to schedule packet transmission based on QoS where an application can specify its requirement by quantifying the end-to-end deadline miss raio it can afford. But this approach assumes that bursts between links are independent which is not yet validated. Hence, we will consider the use of Epsilon-Stationary links in our future work. Table 2 shows how the classification of the links can be useful to select links for different application purposes.

5. ALGORITHM

To explain our algorithm we first present some preliminary discussion. In Section 5.1, using a simple example that assumes a single stream we show how to deal with end-to-end latency in the presence of burstiness. In Section 5.2, we then expand the example to multiple streams where both burstiness and interference must be handled. After these preliminary discussions we present the complete algorithm for latency bound generation.

5.1 Dealing with a Single Stream

If there is only one stream in the network and there is no packet loss, then the end-to-end latency bound is the sum of per link latencies in all the intermediate links. This is the theoretical lower bound of end-to-end latency for a sin-

$Bmax$	Links	Application
Small	Stationary, Asymptote-Stationary	High Priority Hard Real Time Applications
Large	Stationary, Asymptote-Stationary	Low Priority Hard Real Time Applications
Small	Epsilon-Stationary	High Priority Soft Real Time Applications
Large	Epsilon-Stationary	Low Priority Soft Real Time Applications
Small/ Large	Non-Stationary	Other Applications

Table 2: Classification of Links for Different Applications

gle stream. But in reality, links are not ideal and packet losses occur in a burst. If a stream has n intermediate links from source to destination and ith intermediate link has burstiness parameters $Bmax = b_i$ and $B'min = b'_i$ $(i = 1, 2, 3, \cdots, n)$, then on ith link we have to allocate $b_i + 1$ time slots for the stream. So the end-to-end latency bound LB is given by Equation (1).

$$LB = \sum_{i=1}^{n} (b_i + 1) \quad (1)$$

Consider the topology in Figure 1. Assume that the links have $Bmax$ and $B'min$ as shown in Table 3.

Now, assume that our SS consists of a single stream S_1 from Table 1 having period 20, starting time slot 1 and it goes from N_1 to N_4 using route $N_1 \rightarrow N_2 \rightarrow N_3 \rightarrow N_4$. So, $SS = \{(S_1, N_1, N_4, RT_1, 1, 20)\}$ where $RT_1 = \{N_1, N_2, N_3, N_4\}$.

Since, $Bmax$ is 2 for $L(1, 2)$, if we allocate $(Bmax + 1)$, i.e., 3 time slots for S_1 at node N_1, then the packet will reach to N_2 even if there is a burst. Similarly, if we allocate 4 time slots at N_2 to transmit over $L(2, 3)$ and 4 time slots at N_3 to transmit over $L(3, 4)$ then it takes only $3 + 4 + 4 = 11$ time slots to ensure that every packet of S_1 will be delivered to its destination within that time even if there is a burst in a number of links. Hence, $LB_1 = 11$. The corresponding scheduling table is shown in Table 4 below where the column represents time slots and the row represents links. It shows which stream will be transmitted at which time slot using which link. For example, S_1 will be transmitted at time slots 1, 2 and 3 using link $L(1, 2)$.

Link	$Bmax$	$B'min$
$L(1, 2)$	2	2
$L(2, 3)$	3	2
$L(3, 4)$	3	3
$L(4, 5)$	3	2
$L(7, 8)$	2	2
$L(17, 18)$	2	3
$L(18, 19)$	1	4

Table 3: $Bmax$ and $B'min$ of Links

Note that multiple time slots are reserved to ensure reliablity. It does not mean that N_1 is transmitting three packets of S_1 in time slots 1, 2 and 3. Rather it means that N_1 will try in these time slots to transmit a packet of S_1 and it will stop as soon as the packet gets in to the next hop node, i.e., N_2. We assume that the radio transceiver supports hardware/software acknowledgements so that the receiver can acknowledge its packet reception immediately to the sender. This is a reasonable assumption based on the radio transceivers available in the market.

Link/TimeSlot	1	2	3	4	5	6	7	8	9	10	11
$L(1, 2)$	S_1	S_1	S_1								
$L(2, 3)$				S_1	S_1	S_1	S_1				
$L(3, 4)$								S_1	S_1	S_1	S_1

Table 4: Scheduling Table with a Single Stream

5.2 Dealing with Multiple Streams

When we have only one stream in the network, even in the presence of burstiness, the scheduling is not complex since the interference is not an issue. No two links can interfere with the transmission of each other in this scenario. But when multiple streams co-exist in the network, we have to take interference into account and ensure that no two interfering links transmit packets at the same time slot causing a pacekt loss. Note that finding the minimum average latency bound of all the streams by scheduling packet transmission considering all possible routes for every streams subject to link interference and link burstiness is a NP-Hard problem since it is analogous to the bin packing problem [5]. Hence we use a greedy solution based on the following principles:

1. Schedule packet transmission up to the LCM of the periods of all the streams. If all the streams are schedulable within the LCM of their periods, we conclude that the stream set is schedulable and we can offer a latency bound for every stream.

2. Address packet loss due to link interference as follows: First, figure out the interference pattern of the network, represented by IM empirically. Then, schedule packet transmission in a way to make sure that no two interfering links 'transmit' packets at the same time slot.

3. Address packet loss due to link burst as follows:

   (a) Allocate $Bmax$+1 contiguous time slots for packet transmission over a link. Note that different links have different $Bmax$. We are assuming that the route is *least-burst-route* as the route is the shortest path having the minimum sum of $Bmax$ from the source node to the destination node.

   (b) While allocating $Bmax + 1$ time slots, overlap at most $B'min$ streams' time slot allocation. The reason is, within a window of $Bmax + B'min$, there are at least $B'min$ good slots for packet transmission, and so, we should be able to transmit at least $B'min$ number of streams. Hence, we allow slots of at most $B'min$ streams to overlap. There are two conditions for it: (i). This overlapping is allowed only when packets are being transmitted over the same link. Note that we

can not overlap time slots for neighboring nodes due to link interference. (ii). While overlapping time slots of multiple streams, no two streams are allowed to be allocated the same $Bmax + 1$ time slots, i.e., complete overlapping is not allowed. The reason is, if two streams S_1,S_2 are allocated the same $Bmax + 1$ time slots, and if we lose $Bmax$ time slots due to a burst, then we can not transmit packets of both S_1 and S_2 in the remaining time slot.

Link/TimeSlot	1	2	3	4	5	6	7	8
$L(1,2)$	S_1	S_1	S_1	S_1				
$L(1,2)$					S_2	S_2	S_2	S_2

Table 5: Schedule without Overlapping

Link/TimeSlot	1	2	3	4	5
$L(1,2)$	S_1	S_1, S_2	S_1, S_2	S_1, S_2	S_2

Table 6: Schedule with Overlapping

To explain why overlapping is important and how it is done, consider the following example. Assume that Stream Set SS consists of two streams S_1, and S_2, both going from node N_1 to N_2 using route $N_1 \rightarrow N_2$ through the link L(1,2). Assume that both streams have period 20 and starting time slot 1. Also assume that $Bmax = 3$ and $B'min = 2$ for the link L(1,2). A schedule without overlap is shown at Table 5 where average Latecy Bound per stream is $(4+8)/2 = 6$ time slots. Compare it with a schedule with overlapping in Table 6 where average Latency Bound per stream is $(4+5)/2 = 4.5$ time slots. Note that we could not do complete overlapping as in Table 7. Because, as it is mentioned earlier, if we lose $Bmax = 3$ time slots due to a burst, then we can not transmit packets of both S_1 and S_2 in the remaining time slot.

Link/TimeSlot	1	2	3	4
$L(1,2)$	S_1, S_2	S_1, S_2	S_1, S_2	S_1, S_2

Table 7: Schedule with Complete Overlapping

Overlapping time slots raises another issue. Since we can transmit only one packet at a time and the packet transmission process is completely deterministic, we have to prioritize the streams to be transmitted in the overlapped time slots. Our *prioritizing rule* works as follows: if multiple streams are scheduled to be transmitted at the same time slot, transmit the not-yet-transmitted stream that has the closest ending time slot. We will see its use in the next example.

To illustrate how many streams can be overlapped consider the following example. Assume that Stream Set SS consists of four streams S_1,S_2,S_3, and S_4 all going from node N_1 to N_2 using route $N_1 \rightarrow N_2$ through the link L(1,2). Assume that all the streams have period 20 and starting time slot 1. Also assume that $Bmax = 2$ and $B'min = 4$ for the link L(1,2). Table 8 demonstrates a schedule that shows within a window of size $Bmax + B'min = 2 + 4 = 6$, we can overlap at most $B'min = 4$ streams. This schedule

Link/TimeSlot	1	2	3	4	5	6
$L(1,2)$	S_1	S_1, S_2	S_1, S_2, S_3	S_2, S_3, S_4	S_3, S_4	S_4

Table 8: Schedule with Maximum Overlapping

will always work if nodes transmit packets according to the prioritizing rule. For example, if packet transmission fails during time slots 1, and 2, packets of streams S_1,S_2,S_3, and S_4 will be transmitted at time slots 3,4,5 and 6, respectively. If packet transmission fails at time slots 2, and 4 then packets of streams S_1,S_2,S_3, and S_4 will be transmitted at time slots 1,3,5 and 6. So, even if packet transmission of any combination of size $Bmax$ fails within a window of $Bmax + B'min$, all the packets of all the streams will get through to the next node if we allocate time slots according to the principles mentioned earlier and nodes transmit packets according to the prioritizing rule as it is proved in the next subsection.

5.3 Correctness Proof

The correctness of our algorithm depends on the following theorem:

Theorem: *If we overlap packet transmissions of at most b' streams in $(b + b')$ time slots, all going through the same link having $Bmax = b$, $B'min = b'$, each stream having $(b+1)$ contiguous time slots allocated without complete overlapping with any other stream, then all of the streams will get through even if there is a burst of at most b time slots (not necessarily contiguous), if we transmit packets according to our prioritizing scheme.*

Proof: The proof is by contradiction. For a contradiction assume that stream S_i could not be transmitted due to a burst. Since we can lose at most b time slots due to a burst, there has to be a time slot, we call "good slot", when there was no burst, but still S_i was not transmitted. The reason for that can only be some other stream S_j was transmitted at that time slot. Note that the good slot can not be the last sending time slot of S_i. Because, at that time slot, steam S_i has highest priority for transmitting packet. So, no other stream S_j can be transmitted at that time slot. So, now we have to consider one remaining possibility. There was a burst at the last sending time slot of S_i and among the previous b slots of S_i, $(b-1)$ slots are gone due to a burst leaving one good slot and at that time slot some other stream S_j was transmitted for the priority scheme. Note that this can not happen either, because as S_j has higher priority than S_i, S_j has started time slots before S_i did, and hence it should be transmitted even before the burst happens. Because, we are overlapping at most b' streams in any time window of $(b+b')$ slots and there has to be at most b' good slots between any consecutive bursts. So, S_j will be transmitted in the good time slots of the previous window and S_i will be transmitted within this time window. So, there is no way S_i will not be transmitted due to a burst.

5.4 Algorithm Details

Algorithm 2 schedules packet tranmission up to the LCM of the periods of the streams based on the principles and conditions stated in Section 5.2. It takes the Stream Set SS of n streams (as defined in section 2), topology, Interference Matrix IM, burstiness parameters $Bmax$, $B'min$ of every link as inputs and returns either *true* if all the streams are schedulable or $false$ if they are not schedulable. If all the

streams are schedulable, then LB_i holds the latency bound of stream S_i. Let L_i be the last allocated time slot for S_i

Algorithm 2 : Scheduler(SS, topology, IM, $Bmax$, $B'min$ of every link)

```
1: n ← |SS|
2: lcm ← LCM(P_1, P_2, P_3, ... P_n)
3: for i ← 1 to n do
4:    L_i ← ST_i − 1,
5:    N_i ← SRC_i,
6:    LB_i ← 0
7: end for
8: for t_1 ← 1 to lcm do
9:    if All the streams are scheduled then
10:      break
11:   end if
12:   for i ← 1 to n do
13:      if S_i is already scheduled then
14:         continue
15:      end if
16:      if t_1 = L_i then
17:         for t_2 ← t_1 + 1 to ST_i + P_i do
18:            (b, b') ← (Bmax, B'min) of link L(N_i, N_{i+1})
19:            if it is possible to allocate (b + 1) contiguous time slots starting from t_2 by following the principles and conditions then
20:               if S_i is not making any overlap with any streams in these time slots AND (t_2 − t_1) > DeferThreshold then
21:                  L_i ← t_2 − 1
22:                  break
23:               else
24:                  Allocate these (b+1) time slots for S_i in N_i
25:                  L_i ← t_2,
26:                  N_i ← N_{i+1}
27:                  if N_{i+1} is the destination node for S_i then
28:                     Consider that S_i is scheduled
29:                     LB_i ← max(LB_i, t_2 + b + 1 − ST_i)
30:                  end if
31:                  break
32:               end if
33:            end if
34:         end for
35:      end if
36:      if mod(t_1, P_i) = 0 then
37:         if S_i is not scheduled yet then
38:            return false
39:         else
40:            ST_i ← ST_i + P_i,
41:            N_i ← SRC_i
42:         end if
43:      end if
44:   end for
45: end for
46: if all the streams are not scheduled within the lcm then
47:   return false
48: end if
49: return true
```

in the scheduling algorithm. So, initially we have $L_i = (ST_i - 1)$ (at line 4) for every stream S_i. We need to adjust the starting time ST_i of stream S_i when its period P_i is over (at line 40) to correcly compute the latency bound LB_i (at line 29). Let, N_i be the node that has received the last transmitted packet of S_i. So, initially $N_i = SRC_i$ (at line 5) which is the source node of stream S_i. Assume that N_{i+1} is the next node to which N_i has to transmit a packet of S_i. Note that N_{i+1} can be easily determined from the route of S_i.

We put $DeferThreshold = 2$ (used at line 20). If you use a higher value for it, the scheduler runs faster, but the generated latency bound may also rise. We are deferring time slot allocation to only those streams that can not take advantage of overlapping time slots with other streams. The reason for deferring time slot allocation is to increase parallelism over the link from N_i to N_{i+1}. This is a kind of lazy approach for allocating, because we are hoping that it is possible that some other streams may show up and can be allocated in these time slots that can exploit parallelism.

Link/TimeSlot	1	2	3	4	5	6	7	8	9	10
$L(1,2)$	S_1	S_1	S_1							
$L(2,3)$				S_1	S_1,S_2	S_1,S_2	S_1,S_2	S_2		
$L(3,4)$									S_1	S_1,S_2
$L(4,5)$										
$L(7,8)$										
$L(8,9)$										
$L(17,18)$	S_4	S_4	S_4							
$L(18,19)$				S_4	S_4					

Table 9: Scheduling Table with Multiple Streams: Part 1

Link/TimeSlot	11	12	13	14	15	16	17	18	19	20
$L(1,2)$										
$L(2,3)$										
$L(3,4)$	S_1,S_2	S_1,S_2	S_2							
$L(4,5)$				S_2	S_2	S_2	S_2			
$L(7,8)$				S_3	S_3	S_3				
$L(8,9)$								S_3	S_3	S_3
$L(17,18)$	S_4	S_4	S_4							
$L(18,19)$				S_4	S_4					

Table 10: Scheduling Table with Multiple Streams: Part 2

Table 9 and Table 10 show the whole scheduling for the example problem we are working with from Section 2 and B, B' of Table 3. Here S_1 and S_2 share time slots at links $L(2,3)$ and $L(3,4)$. The latency bound of the streams is shown in Table 11.

Stream	Latency Bound
S_1	12 i.e. $\max(12-1+1,0)$
S_2	17 i.e. $\max(17-1+1,0)$
S_3	20 i.e. $\max(20-1+1,0)$
S_4	5 i.e. $\max((5-1+1),(15-11+1),0)$

Table 11: Latency Bound for Each Stream

Our algorithm exploits parallelism in two ways. The first one is, multiple streams can be transmitted at the same time

if there is no interference in their transmission as it is seen in S_1 and S_4 at time slots 1, 2 and 3 in Table 9. The second way is, when two streams are going over the same link, they can share at most $Bmax$ time slots as it is observed between S_1 and S_2 in time slots 5, 6 and 7 in Table 9.

The running time of the algorithm is $O(LP)$ where L is the LCM of the periods of the streams and P is the sum of the periods of the streams. The reason is, the for loop at line 8 takes $O(L)$ time and the for loops at lines 12 and 17 together takes $O(P)$ time. Note that $O(LP)$ depends mainly at the periods of the streams, and it can be exponential if the streams have periods that are prime numbers.

6. EVALUATION

In this section we evaluate our algorithm in terms of end-to-end deadline miss ratio and latency bound. At first we describe the experimental setup. Then we describe how we measure burstiness parameters $Bmax$ and $B'min$ of every link and the interference matrix, IM. Then the effect of $Bmax$ and $B'min$ to end-to-end deadline miss ratio and latency bound is evaluated.

6.1 Experimental Setup

As mentioned in Section 4, we run a 21 day long experiment to understand how burstiness affects packet transmission in an indoor testbed. This testbed consists of 48 TMotes that use the ChipCon CC2420 radio. These nodes are deployed on walls and ceilings of a building, as shown in Figure 6.

6.2 Measuring Burstiness

To understand the effect of link burstiness on packet transmission and to calculate the model parameters $Bmax$ and $B'min$ of every link, we transmit 3,600,000 packets per link. The packet transmission rate is around 200 packets per second where every node tries to transmit the next packet as soon as possible with a zero inter packet interval. To avoid possible collisions and interference, we allow only one node to transmit packet at a time. There are 48 nodes, each node takes turn for packet transmissoin and at each turn a node transmits around 1200 packets continuously. We are supposed to transmit $21 \times 24 \times 60 \times 60 \times 200/48 = 7560000$ packets per link in 21 days with the specified packet transmission rate. But we could only transmit 3,600,000 packet per link, because it takes time to fetch the sequence number of the received packets from all other nodes when a node finishes its turn of packet transmission. After collecting sequence numbers of received packets on every link we have a long data trace that is used to calculate model parameters $Bmax$ and $B'min$ of every link using algorithm 1. The spatial distribution of $Bmax$ of the links at the testbed is shown in Figure 8, where light zones represent links having low $Bmax$, and dark zones represent links having high $Bmax$. We find that most of the dark zones fall within places where peoples' movement varies a lot, e.g., stairs, restrooms, copy-room, conference rooms, and kitchen. There may be other factors involved, e.g., wind, distance to Wi-Fi access point, temperature, distance between links, etc. We need to perform further experiments to determine the most dominant factor that affects $Bmax$.

6.3 Measuring Interference

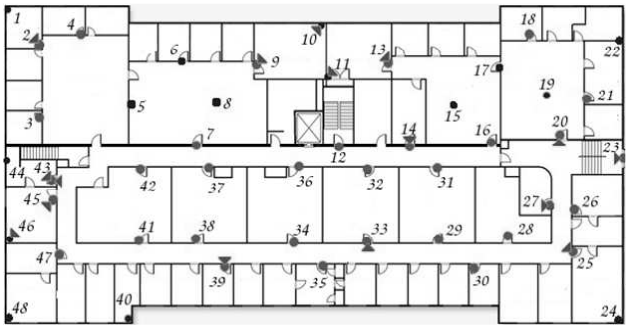

Figure 6: Testbed Layout

The external inteference, e.g. interference caused by WiFi networks, is captured through the $Bmax$ and $B'min$ parameters. To address internal interference, i.e. interference caused by packet transmission through neighboring links at the same time, we define a $k \times k$ interference matrix, IM for a network of k links that specifies which links potentially interfere with others:

$$IM(i,j) = \begin{cases} 1 & \text{if link } L_i \text{ and link } L_j \text{ are in interference range,} \\ 0 & \text{otherwise.} \end{cases} \quad (2)$$

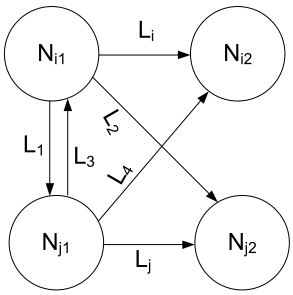

Figure 7: Measuring IM

If $IM(i,j) = 1$, then scheduler will make sure that two packets are not scheduled for transmission over links L_i and L_j at the same time. The computation of IM is based on Packet Reception Ratio (PRR), and PRR is computed from the same data trace that has been used to measure the burstiness parameters. We explain how IM is computed by using Figure 7. Assume that link L_i spans from node N_{i1} to node N_{i2} and link L_j spans from node N_{j1} to node N_{j2}. We compute PRR of every link and then set $IM(i,j)$ to 1 if any of the links L_1, L_2, L_3 or L_4 have a PRR greater than PRR_t, a threshold. We set PRR_t to 0.3 in our experiment. This is a conservative model that hurts the latency bound, but improves the miss ratio.

6.4 Effect of $Bmax$

In this section we describe the effect of $Bmax$ on end-to-end deadline miss ratio and latency bound. In our problem definition described in Section 2 the streams did not have any deadline associated with them and we define the deadline of the streams to be their generated latency bound. We

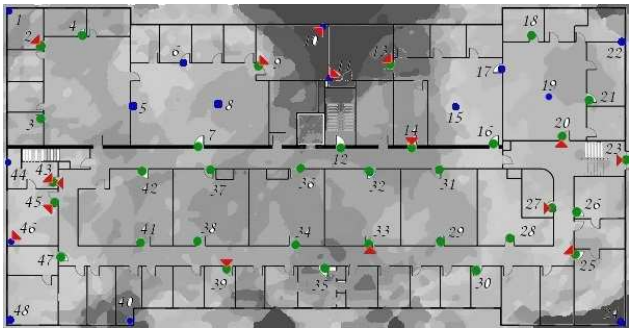

Figure 8: Spatial Distribution of $Bmax$

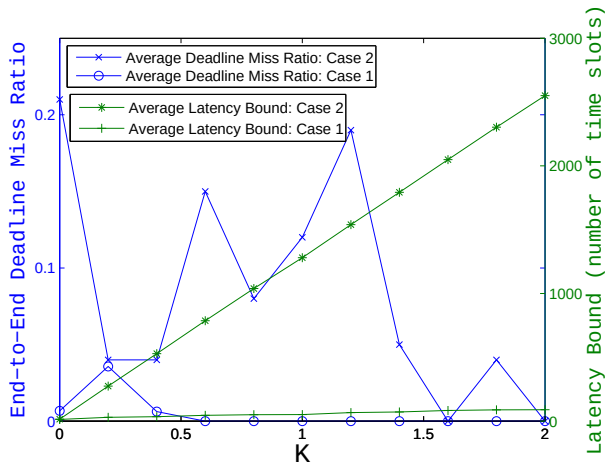

Figure 9: Effect of $Bmax$ on End-to-end Deadline Miss Ratio and Latency Bound

evaluate it on the testbed. We compute the burstiness parameters $Bmax$,$B'min$ and the interference matrix IM of the actual testbed network exactly the way specified in Sections 6.2 and 6.3.To ensure packet transmission experiences the same burst-behavior, at this experiment we maintain the same packet transmission rate of around 200 packets per second with a zero inter packet interval that we use to measure link burstiness as described in Section 6.2.

To disable the effect of $B'min$ we select $B'min$ to 1 for all links. We define a multiplying factor K to demonstrate a trade off between latency bound and end-to-end deadline miss ratio and instead of allocating $Bmax+1$ time slots per link, we are allocating $Bmax \times K + 1$ time slots for different values of K ranging from 0 to 2.

We run the experiment by considering two cases. In case 1, we use the whole testbed for evaluation. This case represents an actual deployed system. But in case 2, we assume that we don't have the top 25% links and we are forced to use some non-stationary links. This case represents another system where links are not as good. The criteria for discarding the top 25% links is $Bmax$ that assumes that the smaller the $Bmax$, the better the link is. Since we are not using the top 25% of the links in case 2, the workloads for these two cases are different. But for each case, for each value of K, we generate 10 different workloads and the average values are plotted in Figure 9.

To generate a workload, we randomly select 10 pairs of nodes as sources and destinations to generate 10 random streams, and choose the route of those streams as the shortest path from the source node to the destination node having minimum sum of $Bmax \times K+1$ time slots. The starting time for every stream is set to its 1st time slot and period is randomly selected from a pool of even numbers up to 800 time slots for case 1 and 6000 time slots for case 2. If we consider a packet transmission rate of 200 packets per second, and allocate 5 msec per time slot, then the generated latency bound at K=1 is $51 \times 5 = 255$ msec for case 1, which is practical for the implementation of control loops in factory automation.

To time synchronize the nodes, we use a RBS style synchronization technique [7]. Since nodes may experience clock drift, to keep the nodes synchronized over time we need to allocate time slots for periodic broadcasting of the time-synchronization message.

Figure 9 shows the effect of $Bmax$ on latency bound. From the figure we observe that as K increases, the average latency bound increases linearly for both cases. The reason is, as K increases, $Bmax$ of all the links increases linearly and since $B'min$ is one for all the links, there is no overlapping of time slots to reduce latency bound.

Figure 9 also shows the effect of $Bmax$ on end-to-end deadline miss ratio in the testbed. For case 1, with the full testbed, we observe that although there are some fluctuations in the end-to-end deadline miss ratio for different values of K up to 0.6, the end-to-end deadline miss ratio becomes 0 after $K = 0.6$. This implies that when there are many good links in a network, our solution obtains 100% packet delivery within the latency bounds. Surprisingly, we observe that even before $K = 1.0$. Note that our generated latency bound (at $K = 1$) is within 14.2% of the minimum latency (at $K = 0.6$) for which we observe zero end-to-end deadline miss ratio. So, the generated latency bound is relatively tight. Also, if we allocate $0.6 * Bmax + 1$ time slots instead of $Bmax + 1$ time slots, we can save 12.4 % of average latency and can still make all the deadlines. We may not want to set $K < 1$ for hard real time applications, but it can be very useful to control the trade off between end-to-end deadline miss ratio and latency bound for soft real time applications.

For case 2, with the top 25% links removed, we observe a different result. Here, missing of deadlines continues beyond $K \geq 1$ even though we are allocating $Bmax \times K + 1$ time slots. The reason behind this is, links having high $Bmax$ are typically susceptible to the changes in the physical environment as shown in Figure 8 and to accurately characterize these links, empirical data should be collected over more than 21 days. Since are choosing least-burst-route for packet transmission, in case 1, we have sufficient number of links having low $Bmax$ for packet transmission while in case 2, we are forced to choose some links having high $Bmax$, which tend to be Epsilon Stationary links and Non-Stationary links. We can differentiate the links based on 21 days of link characterization as shown in Figure 5, but Epsilon Stationary links having high $Bmax$ require more than 21 days for a complete characterization of the environmental dynamics that affect them. The burst size may become bigger than the measured one no matter how long the empirical characterization is. In this case, we can address the

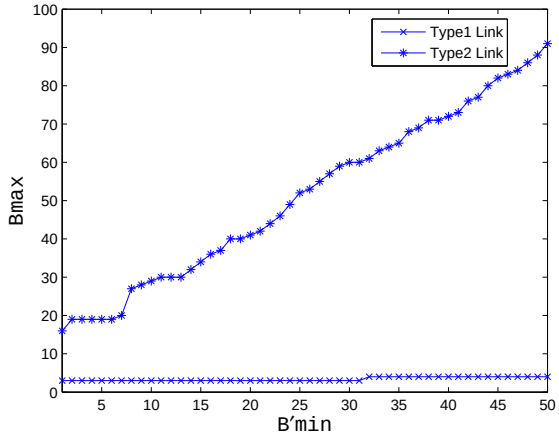

Figure 10: Variation of $Bmax$ for Different $B'min$

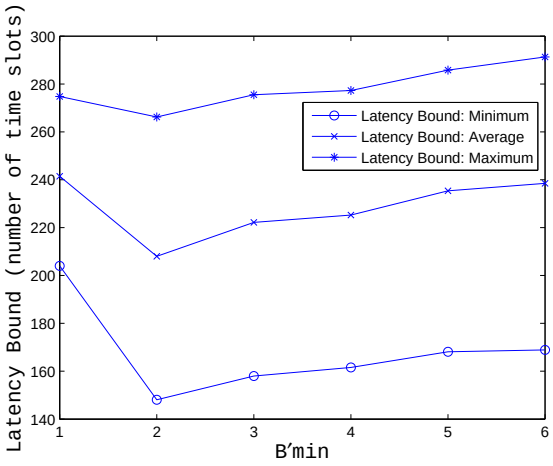

Figure 11: Effect of $B'min$ on Latency Bound

problem with two different approaches: packet recovery and link adaptation as discussed in Section 7.

6.5 Effect of $B'min$

In this section we discuss the effect of $B'min$ on the latency bound. For a particular stream, its latency bound will either increase of remain same if $B'min$ is increased for every link that the stream goes through. The effect depends on the quality of the link. For high quality links, either stationary or asymptote stationary, the response to the increase of $B'min$ is of two types. So, we call these links as Type1 and Type2 links. The response of the two types of links to different values of $B'min$ is shown in Figure 10.

In a Type1 link, as we see from Figure 10, $Bmax$ increases very slowly with the increase of $B'min$. As a result, the latency bound remains the same for some time and increases very slowly for the increase of $B'min$. In the case of a Type2 link, $Bmax$ increases rapidly with the increase of $B'min$ and that's why the latency bound for a particular stream also increases rapidly. The reason behind this behavior depends on the link quality. After transmitting 3,600,000 packets over every link, we compute $Bmax$ for different values of $B'min$ by using algorithm 1. We observe that Type1 links are so good that even if we increase $B'min$, $Bmax$ remains almost the same and increases very slowly with keeping at least $B'min$ number of good slots for packet transmission in every possible window of size $Bmax + B'min$. In the case of Type2 links, as $B'min$ increases, to keep at least $B'min$ number of good slots in every possible window of size $Bmax + B'min$, we need to increase the window size by increasing $Bmax$ rapidly.

However, for a set of streams, the average latency bound may increase, decrease or remain same as a result of the increase of $B'min$ of every link depending on quality of the links, either Type1 or Type2, and spatio-temporal overlapping of the streams. If a large number of streams pass through a dense network with some spatial and temporal overlapping in transmission, we observe a decrease in average latency as $B'min$ increases, but after all the overlapping advantage is exploited, the average latency bound starts to rise again. We observe a similar result with 10 randomly generated streams for the testbed for which average latency bound decreases up to $B'min = 2$ and then it starts to rise again as is shown in Figure 11. We run the experiment 5 times for each value of $B'min$ and the minimum and maximum values of the latency bound of 5 runs are also plotted in the figure. It clearly indicates that we can minimize the average latency bound of a set of streams by an intelligent selection of $B'min$ of the links.

7. FUTURE WORK

In this section, we disscuss about the possible improvements of the scheduling algorithm. Instead of back-to-back retransmission, the scheduler can be smart enough to defer retransmissions a little, or retransmit on another channel, or use an alternate route, or even employ a combination of these based on the link characteristics.

In this paper we set the same $B'min$ on all the links. To minimize latency bound by a better selection of $B'min$ we need to try different values of $B'min$ starting from 1 until the latency bound starts to rise as shown in Figure 11. To set optimal values of $B'min$ to minimize the latency bound, where different links can have different values of $B'min$, we need to use a more sophisticated algorithm e.g. simulated annealing within the scheduler.

The burst-behavior of the wireless links may change due to a change in the physical environment, e.g. node failure, node replacement, or unexpected obstacles. To address this problem we propose two different approaches: packet recovery and link adaptation.

In a packet recovery approach, every node has a queue to hold the un-transmitted packet. The packet can be transmitted later at some other period if there are extra time slots left for packet transmission after all the packets of that period are transmitted successfully. In this way, the packet reaches the destination although it missed its deadline.

In the link adaptation approach, every node keeps a record about the links when it fails to transmit a packet over those links trying all the allocated time slots. After a specific interval, all the nodes report to some base station about the links over which packet tranmission failed. The base station reschedules the packet transmission by doubling the allocated time slots at those links. This approach can be used to dynamically shrink and expand the latency bound

based on network dynamics. We consider the evaluation of these two approaches as future work.

8. DISCUSSION

In this section, we discuss the energy characteristics of our scheduling algorithm. Although the scheduler allocates redundant time slots for packet transmission, since we use least-burst-routing, most of the selected links are very good and hence single packet transmission suffices in most of the cases. From both the transmitter and receiver's point of view, the radio needs to be only used for single packet transmission time in most of the cases, which is optimal. In the case of packet loss, the transmitter and receiver stay on until the packet is received. This energy consumption can only be beaten on the transmitter side by approaches such as [22], where the transmitter turns off after a packet loss. However, that approach pays in terms of longer receiver wake times. Therefore, we expect our approach to have similar overall energy characteristics to [22].

9. CONCLUSIONS

We have presented a new analysis technique that provides exact characterization and classification of the network links subject to link burstiness and interference. The algorithm is novel because it accounts for both link burstiness and interference and offers a schedule for packet transmission that produces an upper bound on the latencies of the streams. We expect that this approach will help enable the use of wireless sensor networks in real-time applications.

10. ACKNOWLEDGEMENT

We would like to thank the anonymous reviewers for their insightful comments. We also would like to thank our shepherd Qing Cao for his guidance.

This work was supported, in part, by NSF Grants CSR-0720640 and CNS-0834299.

11. REFERENCES

[1] Wirelesshart overview. `http://www.hartcomm.org/protocol/wihart/wireless_overview.html`.

[2] P. Agrawal and N. Patwari. Correlated link shadow fading in multi-hop wireless networks. In *Tech Report arXiv:0804.2708v2*, 2008.

[3] J. Broch, D. A. Maltz, D. B. Johnson, Y. Hu, and J. Jetcheva. A performance comparison of multi-hop wireless ad hoc network routing protocols. In *Mobicom '98*.

[4] A. Cerpa, J. L. Wong, M. Potkonjak, and D. Estrin. Temporal properties of low power wireless links: Modeling and implications on multi-hop routing. In *MobiHoc '05*.

[5] E. G. Coffman, Jr., M. R. Garey, and D. S. Johnson. Approximation algorithms for bin packing: a survey. In *Approximation algorithms for NP-hard problems*, 1997.

[6] T. L. Crenshaw, S. Hoke, A. Tirumala, and M. Caccamo. Robust implicit edf: A wireless mac protocol for collaborative real-time systems. *ACM Transactions on Embedded Computing Systems (TECS)*, 2007.

[7] J. Elson, L. Girod, and D. Estrin. Fine-grained network time synchronization using reference broadcasts. In *(OSDI 2002)*.

[8] M. Franceschinis, M. A. Spirito, R. Tomasi, G. Ossini, and M. Pidala. Using wsn technology for industrial monitoring: A real case. In *SENSORCOMM '08*.

[9] G. Hackmann, O. Chipara, and C. Lu. Robust topology control for indoor wireless sensor networks. In *SenSys '08: Proceedings of the 6th ACM conference on Embedded network sensor systems*, 2008.

[10] T. He, S. Krishnamurthy, L. Luo, T. Yan, L. Gu, R. Stoleru, G. Zhou, Q. Cao, P. Vicaire, J. A. Stankovic, T. F. Abdelzaher, J. Hui, and B. Krogh. Vigilnet: An integrated sensor network system for energy-efficient surveillance. *ACM Trans. Sen. Netw.*, 2006.

[11] T. He, J. A. Stankovic, C. Lu, and T. F. Abdelzaher. Speed: A stateless protocol for real-time communication in sensor networks. In *ICDCS '03*.

[12] I.-H. Hou, V. Borkar, and P. R. Kumar. A theory of qos for wireless. In *Infocom 2009*.

[13] D. B. Johnson and D. A. Maltz. Dynamic source routing in ad hoc wireless networks. In *Mobile Computing*, 1996.

[14] H. Li, P. Shenoy, and K. Ramamritham. Scheduling messages with deadlines in multi-hop real-time sensor networks. In *RTAS '05*.

[15] S. Lin, J. Zhang, G. Zhou, L. Gu, T. He, and J. A. Stankovic. Atpc: Adaptive transmission power control for wireless sensor networks. In *ACM SenSys '06*.

[16] S. Lin, G. Zhou, K. Whitehouse, Y. Wu, J. A. Stankovic, and T. He. Towards stable network performance for wireless sensor networks. In *IEEE RTSS '09*.

[17] C. L. Liu and J. W. Layland. Scheduling algorithms for multiprogramming in a hard-real-time environment. *J. ACM*, 1973.

[18] C. Lu, B. Blum, T. Abdelzaher, J. Stankovic, , and T. He. Rap: a real-time communication architecture for large-scale wireless sensor networks. In *RTAS '02*.

[19] D. Lymberopoulos, Q. Lindsey, and A. Savvides. An empirical analysis of radio signal strength variability in ieee 802.15.4 networks using monopole antennas. In *ENALAB Technical Report 050501, EWSN 2006*.

[20] V. Raghunathan, S. Ganeriwal, C. Schurgers, and M. B. Srivastava. Energy efficient wireless packet scheduling and fair queuing. In *ACM Transactions on Embedded Computing Systems*, 2004.

[21] T. Rusak and P. Levis. Burstiness and scaling in low power wireless simulation. In *MobiCom'08*.

[22] K. Srinivasan, M. A. Kazandjieva, S. Agarwal, and P. Levis. The β-factor: Measuring wireless link burstiness. In *SenSys '08*.

[23] A. Willig. How to exploit spatial diversity in wireless industrial networks. In *IFAC Annual Reviews in Control*, 2008.

[24] G. Zhou, T. He, S. Krishnamurthy, and J. A. Stankovic. Impact of radio irregularity on wireless sensor networks. In *ACM MOBISYS 2004*.