A System for Bridge Monitoring in Railway

Paper Field: Infrastructure & Foundation

Hamid Reza Hafeznezami[١]
Majid Bayat[٢]

ABSTRACT

Railway systems are critical in many regions, and can consist of several tens of thousands of bridges, being used over several decades. It is critical to have a system to monitor the health of these bridges and report when and where maintenance operations are needed. This paper presents A System for Bridge Monitoring in Railway. The design is driven by two important factors: application requirements, and detailed measurement studies of several pieces of the architecture. In comparison with prior bridge monitoring systems, these contributions are three-fold. First, it designed a novel event detection mechanism that triggers data collection in response to an oncoming train. Next, it employs a simple yet effective multi-channel data transfer mechanism to transfer the collected data onto a sink located on the moving train. Third, this architecture is designed with careful consideration of the interaction between the multiple requisite functionalities such as time synchronization, event detection, routing, and data transfer.

Keywords

Event detection, Mobile data transfer, Bridge Surveying, Wireless sensor applications.

1. INTRODUCTION

Railway systems are a critical part of many nations' infrastructure. For instance, Iranian Railways is one of the largest enterprises in the world. And railway bridges form a crucial part of the system.
In Iran, there are a lot of bridges spread over a large geographical area. Most of these bridges are over 10 years old. It is not uncommon to hear of a major accident every few years due to collapse of a bridge. An automated approach to keeping track of bridges' health to learn of any maintenance requirements is thus of utmost importance. In this paper, we present a system for long term railway bridge monitoring. Two factors guide the design of this system. (1) Given the huge number of existing bridges that need to be monitored, it is important that any solution to the problem should be easy to deploy. (2) Next, since technical expertise is both difficult to get and expensive on field, it is equally

[1] . **Production Engineering Department, Mechanical expert, Irankhodro Rail Transport Industries Co.(IRICO), Sarir Building, Tehran-Karaj highway, Tehran, IRAN, Tel: +98-21-44182131~7, Fax: +98-21-44182130, hafeznezami@iri.co.ir**

[2] . **Production Engineering Department, Electrical & Electronic Processes Chief, Irankhodro Rail Transport Industries Co.(IRICO), Sarir Building, Tehran-Karaj highway, Tehran, IRAN, Tel: +98-21-44182131~7, Fax: +98-21-44182130, bayat@iri.co.ir**

important that the deployments require minimal maintenance. To facilitate ease of deployment, we choose to build our system based on battery operated *wireless* sensor nodes. This system consists of several tens to hundreds of such nodes equipped with accelerometers, spread over the multiple spans of the bridge. The use of wireless transceivers and battery eliminates having to lay cable to route data or power (tapped from the 25 KV overhead high voltage line if available) to the various sensors that are spread about on the bridge. Cables and high voltage transformers typically need special considerations for handling and maintenance: safety, weather proofing, debugging cable breaks, etc.; the use of wireless sensor nodes avoids these issues.
A significant challenge which arises in this context is the following. The nodes need to sleep (turn off radio, sensors, etc) most of the time to conserve power. But they also need to be ready for taking accelerometer measurements when there is a passing train. This system employs a novel event detection mechanism to balance these conflicting requirements. The event detection mechanism consists of a beaconing train and high gain external antennae at designated nodes on the bridge that can detect the beacons much before (30s or more) the train approaches the bridge. This large guard interval permits a very low duty cycle periodic sleep-wake up-check mechanism at all the nodes. It uses a combination of theoretical modeling and experimentation to design this periodic wakeup mechanism optimally. On detecting a train, its nodes collect vibration data. The collected data then has to be transferred to a central location. This data will be used for offline analysis of the bridge's current health as well as for tracking the deterioration of the bridge structure over time (several months/years). For this, it uses an approach quite different from other sensor network deployments: it uses the passing trains themselves for the data transfer. The data transfer mechanism is also activated through the same event detection mechanism as for data collection. A very significant aspect of the mobile data transfer model is that it allows us to break-up the entire system of sensor nodes (of the order of few hundred nodes) into multiple *independent* and much smaller networks (6-12 nodes each). This greatly simplifies protocol design, achieves good scalability and enhances performance.
In overall architecture, apart from event detection and mobile data transfer, two other functionalities play important support roles: time synchronization and routing. Time synchronization is essential both for duty-cycling as well as in the analysis of the data (correlating different sensor readings at a given time). Routing forms the backbone of all communication between the nodes. These four functionalities are all inter-dependent and interfacing them involves several design choices as well as parameter values. It designs this with careful consideration of the application requirements as well as measurement studies. In comparison with prior work in structural monitoring, its contributions are three-fold: (a) a novel event detection mechanism, (b) a detailed design of mobile data transfer, and (c) the tight integration of the four required functionalities. While the notion of mobile data transfer itself has been used in prior work, its integration with the rest of the bridge monitoring system, and careful consideration of interaction among various protocol functionalities, are novel aspects of our work.. In prototype, it uses sensor nodes which have an 8MHz MSP430 processor and the 802.15.4 compliant CC2420 radio, operating in the 2.4GHz band. This prototype also extensively uses external high-gain antennas connected to the motes. Although this design is more or less independent of the choice of accelerometers, it is worth noting that it uses the MEMS-based ADXL 203 accelerometer in our prototype. Estimates based on measurements using this prototype indicate that the current design of this system should be deployable with maintenance requirement only as infrequent as once in 1.5 years or so (with 4 AA batteries used at each node).Though this design has focused on railway bridges so far, we believe that the concepts behind this system will find applicability in a variety of similar scenarios such as road bridge monitoring, air pollution monitoring, etc. The next section provides the necessary background on bridge monitoring and deduces the requirements for system design. Subsequently, Sec. 3 describes the overall this system architecture. Then, Sec. 4, Sec. 5, Sec. 6, and Sec. 7 present the detailed design of the four main modules of this system respectively: event detection, time synchronization, routing, and mobile data transfer. Sec. 8 highlights our contributions prior work in this domain. We present further points of discussion in Sec. 9 and conclude in Sec. 10.

2. BACKGROUND ON BRIDGE MONITORING

In this section, we provide a brief background on bridge monitoring.

General information on bridges: A common design for bridge construction is to have several spans adjoining one another .Depending on the construction, span length can be anywhere from 30m to about 125m. Most bridges have length in the range of a few hundred meters to a few km.
What & where to measure: Accelerometers are a common choice for the purposes of monitoring the health of the bridges . We consider the use of 3-axis accelerometers which measure the fundamental and higher modal frequencies along the longitudinal, transverse, and vertical directions of motion. The placement of the sensors to capture these different modes of frequencies as well as relative motion between them is as shown in Fig. 1.

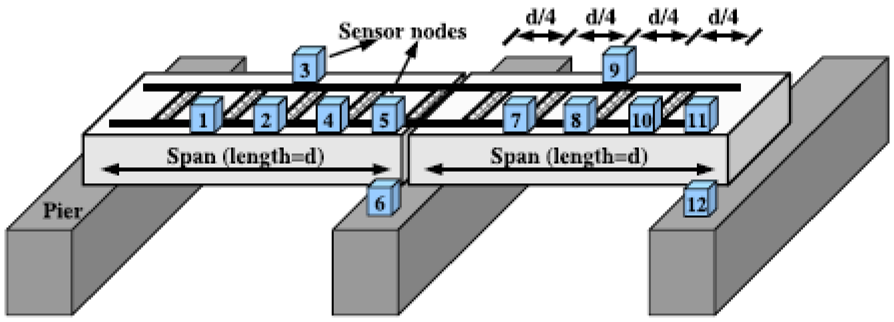


Figure 1: Spans on a bridge

The data collected by the sensors on each span are correlated since they are measuring the vibration of the same physical structure. In some instances of bridge design, two adjacent spans are connected to a common anchorage, in which case the data across the two spans is correlated. For data collection, it defines the notion of a *data-span* to consist of the set of sensor nodes whose data is correlated. A data-span thus consists of nodes on one physical span, or in some cases, the nodes on two physical spans. An important point to note here is that collection of vibration data across different data-spans are independent of each other i.e. they are not physically correlated.
When, how long to collect data: When a train is on a span, it induces what are known as *forced vibrations*. After the train passes the bridge; the structure vibrates freely (*free vibrations*) with decreasing amplitude, until the motion stops. Structural engineers are mostly interested in the natural and higher order modes of this free vibration as well as the corresponding damping ratio. Also of interest sometimes is the induced magnitude of the forced vibrations. For both forced as well as free vibrations, it collects data for a duration equivalent to about five time periods of oscillation.
The frequency components of interest for these structures are in the range of about 0.25 Hz to 20 Hz . For 0.25 Hz, five time periods is equivalent to 20 seconds. The total data collection duration is thus about 40 seconds (20 seconds each for forced and free vibrations).
Quantity of data: As mentioned earlier, each node collects accelerometer data in three different axes (x, y, z). The sampling rate of data collection is determined by the maximum frequency component of the data we are interested in: 20 Hz. For this, it needs to sample at least at 40 Hz. Often oversampling (at 400 Hz or so) is done and the samples averaged (on sensor node itself before transfer) to eliminate noise in the samples. But the data that is finally stored / transmitted would have a much smaller sampling frequency which we set to 40Hz in our case. Each sample is 12-bits (because of use of a 12 bit Analog-to-Digital converter). The total data generated by a node can be estimated as: $3channels \times 12bits \times 40Hz \times 40sec = 57.6Kbits$. There are an estimated 6 sensor nodes per span, and a maximum of 12 nodes per data span. Thus the total data it has per data-span per data collection cycle is a maximum of $57.6 \times 12 = 691.2Kbits$.
Time synchronization requirement: Since the data within a data-span are correlated, it needs time synchronization across the nodes, to time-align the data. The accuracy of time synchronization required is determined by the time period of oscillation above, which is minimum for the highest frequency component present in that data i.e. 20 Hz. For this frequency, the time period is 50ms, so a synchronization accuracy of about 5ms (1/10 of the time period) should be sufficient. Note that this is of much coarser granularity than what is typically described in several time synchronization protocols.

3. DESIGN OVERVIEW

With the application details as given above, the various components of the design are closely inter-related. The prime goal in this design is to have a system which requires minimal maintenance. This translates to two implications. (1) Once installed, the system should be able to run as long as possible without requiring battery replacements. (2) The data collected should be made available from remote bridges to a central data repository where it can be analyzed, faults detected and isolated. Important questions in this context are:

• How does it balance the requirement for duty cycling with the fact that train arrivals are unpredictable?
• How does it transfer the data from the place of collection to a repository?
• How can it achieve scaling, for potentially long bridges?
• What are going to be the inter-dependencies among system components, and how do we re solve them?

Event detection: it balances the requirements of having to duty cycle, and being ready when a train arrives, through its event detection mechanism. It uses the fact that significant radio range is achievable with the 802.15.4 radios, on using external antennas. An 802.15.4 node on the train beacons as it arrives, and is detected several tens of seconds in advance (before the train is on the span) by nodes on the span. This enables a periodic sleep/wakeup mechanism for the nodes on the bridge.

Mobile data transfer: it uses the passing train itself for transferring the data collected. The data is then ultimately delivered to a central repository. This could be done, an Internet connection available at the next major train station. The same event detection used for data *collection* is also used to trigger the data *transfer* to a moving train. A subtle point to note here is that the data collected in response to a train is actually conveyed to the central repository via the *next* oncoming train.

Data span as an independent network: One fundamental design decision in this system is to treat each data-span as an *independent* network. This is possible primarily due to the fact that the data from each data-span is independent physically (different physical structures). This also fits in well with its mobile data transfer model. The alternative here is to treat the entire bridge (including all data spans) as a single network. It rejected this approach since there is no specific reason for the data-spans to know about one another or inter-operate in any way. Having a smaller network simplifies protocol design, and enables much better performance. A designated node on each span *gathers* all the data collected by the different sensor nodes on that span. It then transfers this data onto the moving train. It makes different spans operate on different independent channels, so that the transfer on each span can proceed simultaneously and independently.

Inter-dependence challenges: The event detection as well as data transfer bank on two underlying mechanisms: time synchronization and routing. So there are four main functionalities this system: (a) event detection coupled with periodic sleep/wake-up, (b) mobile data transfer, (c) time synchronization, and (d) routing. In this context, several non-obvious questions arise:

• What protocols should it use for time synchronization and routing?
• More importantly, how should these two interact with any duty cycling?
– Should routing be run for each wake-up period, each time a node wakes up? Or should it be run periodically, across several wake-up periods?
– Similarly, when exactly should time synchronization be run? How does it balance between synchronization overhead and having a bound on the synchronization error?
• Also important is the interaction between routing and time synchronization. Which functionality should build on the other? Can time synchronization assume routing? Or should routing assume time synchronization?

Approach to time synchronization: it requires time synchronization for two things: for the periodic sleep/wake-up mechanism, and for time-aligning the sensor data from different nodes.
Its periodic sleep/wake-up has a period of the order of 30-60s, with wake-up duration of about 200ms. And it shows that it can have light-weight time synchronization Mechanism run during every wake-up duration, at no extra cost. In the time-period between two wake-up durations, of about a minute, the

worst case clock drift can be estimated. The work in reported a worst-case drift of about 20ppm for the same platform as it's. This means a maximum drift of 1.2ms over 60s. This is negligible as compared to its wake-up duration, and hence exact drift estimation is unnecessary. With respect to application too, the time synchronization requirement is not that stringent. Recall from Sec. 2 that it requires only about 5ms or less accuracy in synchronization. So this too does not require any drift estimation. Its approach to time synchronization is simple and efficient, and is in contrast with protocols in the literature such as FTSP. FTSP seeks to estimate the clock drift, and synchronize clocks to micro-second granularity. Due to this, it necessarily takes a long time (of the order of a few minutes).
Approach to routing: The first significant question which arises here is what is the expected stability of the routing tree; that is how often this tree changes. This in turn depends on link stability. For this, we refer to an earlier study in , where the authors show the following results, on the same 802.15.4 radios as used in our work.
(1)When it operates links above a certain threshold RSSI (received signal strength indicator), they are very stable, even across days.
(2) Below the threshold, the link performance is unpredictable over small as well as large time scales (few sec to few hours).
This threshold depends on the expected RSSI variability, which in turn depends on the environment. In practice, in mostly line of- sight (LOS) environments, operating with a RSSI variability margin of about $10dB$ is safe. So, given that the sensitivity of the 802.15.4 receivers is about $-90dBm$, having an RSSI threshold of about $-80dBm$ is safe. With such a threshold based operation, in, it is observed that link ranges of a few hundred meters are easily achievable with off the- shelf external antennas.

4. EVENT DETECTION IN THIS SYSTEM

Event detection forms the core of this system. It is needed since it is difficult to predict when a train will cross a bridge. It needs to go hand-in-hand with a duty cycling mechanism for extending battery lifetime, and thus minimizing maintenance requirements. In the description below, it first assume that the nodes are synchronized. And it assumes that it has a routing tree, that is, each node knows its parent and its children. Once we discuss time synchronization and routing, it will become apparent as to how the system bootstraps itself. At the core of its event detection mechanism is the ability to detect an oncoming train before it passes over the bridge. It seeks to use the 802.15.4 radio itself for this.
Event detection model: Its model for event detection is depicted in Fig. 2. It has an 802.15.4 node in the train which beacons constantly. Let *Dd* denote the maximum distance from the bridge at which beacons can be heard from the train at the first node (node-1 in Fig. 1), if it were awake. Assume for the time being that the node can detect this instantaneously, as soon as the train comes in range; it shall later remove this assumption. It denote by *Tdc* the maximum time available between the detection of the oncoming train, and data collection. Thus *Tdc = Dd/V* where *V* is the speed of the train (assumed constant).

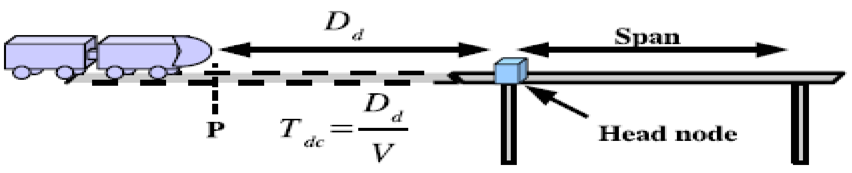


Figure 2: Detecting an oncoming train

In this design, all nodes duty cycle, with a periodic sleep/wake-up mechanism. One node per data-span is designated as the *head* node. This is typically the node which the train would pass first . This head node has the responsibility of detecting an oncoming train. During its wake-up period, if it detects a beacon from a train, it sends out a *command* to the rest of the nodes in the network to remain awake (and not go back to sleep), and start collecting sensor data. So the other nodes have to listen for this command during the time they are awake.

4.1 Radio range experiments

The distance *Dd* essentially captures the distance at which beacons sent from the train can be received at the head node. Measurements in indicate that if it uses external antennas connected to 802.15.4 radios, it can achieve radio ranges of a few hundred meters in line-of-sight environments. However, does not consider mobile 802.15.4 radios. Hence it performed a series of careful experiments with one stationary node and one mobile node4. We note that we usually have about a 1km *approach* zone ahead of a bridge. This is straight and does not have any bends. This is true for most bridges, except in hilly regions. We used a 900m long air-strip. We mounted the stationary node on a mast about 3m tall. We placed the mobile node in a car, and connected it to an antenna affixed to the outside of the car at a height of about 2m. Both nodes were connected to 8dBi Omni-directional antennas. The mobile node beacons constantly, every 10ms. It starts from one end of the air-strip, accelerates to a designated speed and maintains that speed (within human error). The stationary node is 100m away from the other end (so that the car can pass the stationary node at full speed, but still come to a halt before the air-strip ends). For each beacon received at the receiver, we note down the sequence number and the RSSI value. Fig. 3 shows a plot of the RSSI as a function of the distance of the mobile sender from the receiver.

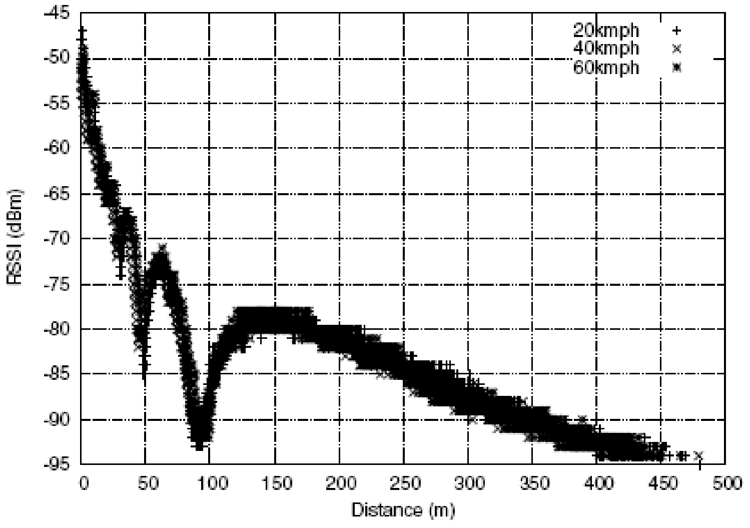


Figure 3: RSSI vs. distance betn. Sender & receiver

An immediate and interesting observation to note in Fig. 4 is the pattern of variation in the RSSI as we get closer to the stationary node, for all mobile speeds. Any RSSI variations observed are generally attributed to unpredictable environmental aspects. The pattern is entirely predictable: these are due to the alternating constructive & destructive interference of ground reflection which happens at different distances. The exact distance at which this happens depends on the heights of the sender/receiver from the ground. Such variations can be eliminated by using diversity antennas, but there are some hardware does not have such a facility. We observe from Fig. 4 that it start to receive packets when the mobile is as far away as 450m, and this is more or less independent 4We intentionally describe these experiments here, and not in a later section, since the results of these experiments were used to drive design in the first place. It must be a person sitting in the car press the *user* button whenever the car passed a 100m mark; this gives us a mapping between the mote's timestamp and its physical position. The RSSI measurements versus distance also have implications for the link range in the (stationary) network on the bridge. If we follow a threshold-based link model, with a threshold of $-80dBm$, as described earlier, we can have link ranges as high as 150-200m.
Now, it is possible to detect transmissions from an 802.11 sender at an 802.15.4 receiver since they operate in the same frequency (2.4GHz). For this, it can use The CCA (Clear Channel Assessment) detection at the 802.15.4 receiver.

4.2 Frontier nodes

One other mechanism it proposes to further increase *Tdc* is the use of *frontier* nodes. Frontier nodes are essentially nodes placed upstream of the sensor network (upstream with respect to the direction of the train). These nodes do not participate in any data collection, but only serve to detect the oncoming train much earlier.

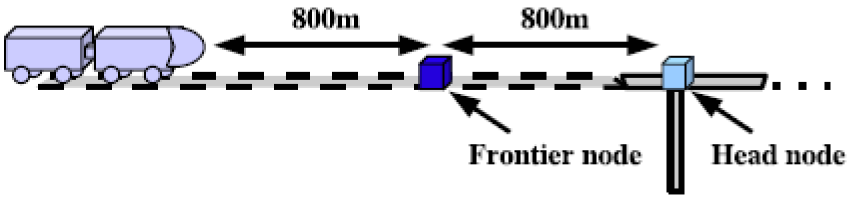


Figure 4: Using frontier nodes to increase *Tdc*

An example in Fig.4 illustrates the use of frontier nodes. *Tdc* is effectively doubled. Note that depending on the timing, it could be the case that the head node directly learns of train arrival, instead of the frontier node telling it. A relevant alternative to consider here, to extend network lifetime, is to simply have additional battery installed at each of the nodes instead of having additional infrastructure in terms of a frontier node. Note however that adding a frontier node improves the battery life of *all* nodes uniformly by decreasing the duty cycle, and hence is likely more beneficial.
It is possible to extend the concept of frontier nodes to have more than one frontier node to detect the oncoming train even further earlier. But the incremental benefit of each frontier node would be lesser. Further, each frontier node also adds additional maintenance issues. However, the use of directional antennas on a moving train is likely to be an engineering challenge. While we may use a directional antenna on a frontier node or on the head node, this would then necessitate the use of at least two antennas, one for either direction.

5. TIME SYNCHRONIZATION

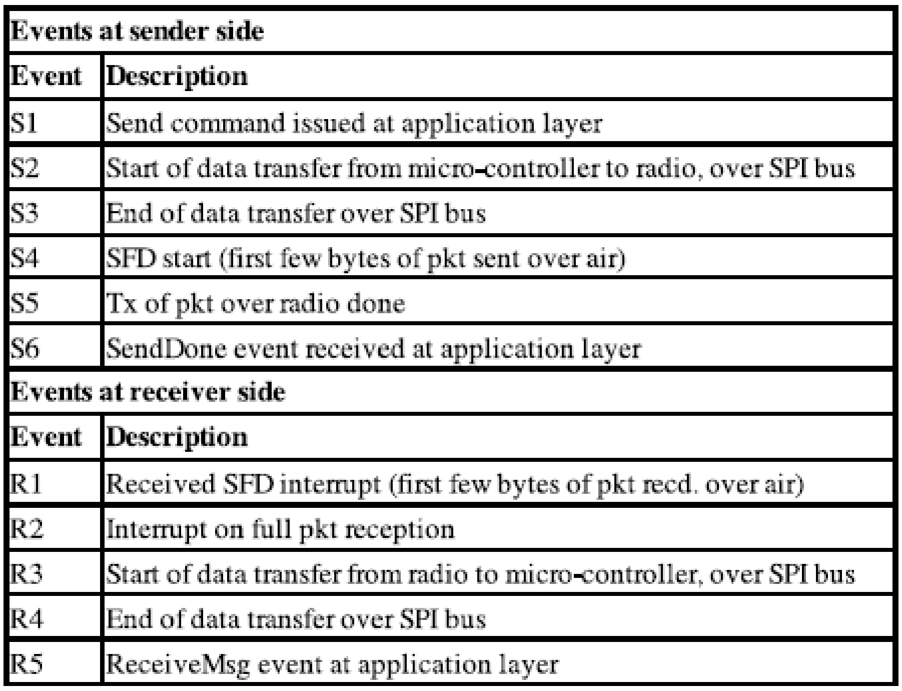
The next important aspect it looks at design is the time synchronization. This aspect is related close to the periodic sleep/wake-up and event detection: two of the parameters in event detection, *Tpc* and *T*Δ, are both related to time synchronization. There are two separate questions here: *how* to do time synchronization (i.e. the time-sync protocol), and *when* the protocol should be run.

5.1 How to do time synchronization?

When an 802.15.4 node *A* sends a message to node *B*, it is possible for *B* to synchronize its clock to that of *A*. This is explained for the Tmote platform in. Thus intuitively it is possible for the entire network to synchronize itself to the head node's clock when a message goes (possibly over multiple hops) from the head node to the other nodes. Now, in this *command* issue from the head node (Sec. 4) too, the message exchanges involved are exactly the same: a message has to go from the head node to the other nodes. In fact, the same protocol message sequence can be used for synchronization as well as for command issue. Only the content of the messages need to be different: for synchronization we would carry time-stamps, and for command issue, an appropriate byte to be interpreted at the receivers.

The issue of flow control

To gain an in-depth understanding of the various delays in the system during transmission & reception, it conducted the following experiment. It sent a sequence of packets, of a pre-determined size, from one mote to the other. It had sufficient gap (over 100 ms) between successive packets to ensure that no queuing delays figure in its measurements. It also disabled all random back offs.
It recorded time-stamps for various events corresponding to each packet's transmission as well as reception. These events, termed *S*1... *S*6 at the sender and *R*1 ...*R5* at the receiver, are listed in Tab. 1. For an event *Si* or *Ri*, denote the time-stamp as *t* (*Si*) or *t* (*Ri*) respectively. Tab. 2 tabulates the various delays corresponding to these events. The different rows in Tab. 2 correspond to experiments with packet sizes of 44, 66, and 88 bytes (including header overheads) respectively. The delay values in the table are the averages over several hundred packets; the variances were small and hence omit it. Given the average delay value, it then calculates the speed of SPI/radio transfer; these values are also shown in the table. 6 It used 3 transmissions: 1 original + 2 retransmissions in its implementation.



Events at sender side	
Event	Description
S1	Send command issued at application layer
S2	Start of data transfer from micro-controller to radio, over SPI bus
S3	End of data transfer over SPI bus
S4	SFD start (first few bytes of pkt sent over air)
S5	Tx of pkt over radio done
S6	SendDone event received at application layer
Events at receiver side	
Event	**Description**
R1	Received SFD interrupt (first few bytes of pkt recd. over air)
R2	Interrupt on full pkt reception
R3	Start of data transfer from radio to micro-controller, over SPI bus
R4	End of data transfer over SPI bus
R5	ReceiveMsg event at application layer

Table 1: Events recorded to measure various delays

It notes that the radio speed is close to the expected 250Kbps at both the sender and the receiver, for all packet sizes. (It is slightly higher than 250Kbps at the sender side, because its measurement of the $[t\,(S5) - t\,(S4)]$ delay marginally underestimates the actual delay on air.) However, a significant aspect it notes is that the SPI speed at the receiver is much lower, only about 160kbps, which is much slower than the radio! This means that packets could queue up at the radio, before getting to the processor. .

5.2 When to do time synchronization?

We now turn to the question of *when* it should run the above time-sync/command protocol. The command to collect data is issued only when a train is detected. With respect to the synchronization protocol, there is the question of how often it should synchronize. It just runs the synchronization mechanism during each wake-up period. This does not incur any additional overhead since anyway all nodes are awake for $Tw = T\Delta + Tpc$, in expectation of a potential command from the head node. And if nodes are awake and listening, the power consumption in the CC2420 chips is the same as (in fact slightly higher than) that of transmitting.

It can now estimate $T\Delta$ too. It is the sum of the possible clock drift in one check cycle (Tcc), and $Terr$, the error in the synchronization mechanism. It estimated $Tdc = 36s$ in Sec. 4.1. Since $Tcc < Tdc$, the worst case $Tdrift$ can be estimated as $20\times10-6 \times 36s = 0.72ms$.

6. ROUTING

It has noted above that both the event detection (command) mechanism and the time-sync protocol depend on the existence of a routing tree rooted at the head node. The crux of its routing mechanism is the fact that it uses stable links.

Routing phases: In designing the routing, it makes the design decision of using a centralized routing approach, with the head node controlling decisions. It has the following simple steps in routing.

(1) *Neighbor-discovery phase:* The head node initiates this phase, by periodically transmitting a HELLO. Nodes which hear the HELLO in turn periodically transmit a HELLO themselves. After some time, each node learns the average RSSI with its neighbors', which it terms the link-state.

(2) *Tree construction phase:* The head node now starts constructing the routing tree. The construction goes through several stages, with each stage expanding the tree by one hop. To begin with, the root node knows its own link-state, using which it can decide its one-hop neighbors. Now it conveys to each of these chosen one-hop neighbors that they are part of the network. It also queries each of them

for their link-state. Once it learns their link-state, it now has enough information to form the next hop in the network. And this process repeats until all possible nodes have been included in the network. In each link state, based on the RSSI threshold, as described in Sec. 3, links are classified as good or bad. The root first seeks to extend the network using good links only. If this were not possible, it seeks to extend the network using bad links. Although simple, the above mechanism has two properties essential for us.

(1) The head node *knows the routing tree* at the end of the two phases. This is essential for its time synchronization and command mechanisms (e.g. to send a command to the nodes in the network to start collecting data, after detecting an oncoming train). It is also essential when it is time for the head node to gather all data from the nodes in the network, before transferring it to the train.

(2) More importantly, the head node *knows when the routing protocol has ended operation*. It stress that this is a property not present in any distributed routing protocol in the literature, to our knowledge. And this property is very essential for power efficient operation: once the head node knows that routing has ended, it can then initiate duty cycling in the network. Such interfacing between routing and duty cycling is an aspect which has not really been looked at in-depth in prior work.

Receive synchronization messages for a certain timeout period), and take-over the head's functionality.

Benefits of a centralized approach: A centralized approach has several benefits, apart from the simplicity in design and implementation. For example, recall from Sec. 5 that *Tpc* is proportional to the number of non-leaf nodes in the network. So it needs to minimize the number of non-leaf nodes. A centralized routing approach can optimize this much more easily as compared to a distributed approach. Similarly, any load balancing based on the available power at the nodes is also more easily done in a centralized routing approach.

Routing protocol delay: Since it expect to run the routing protocol only infrequently, the delay it incurs is not that crucial. But all the same, we wish to note that in this prototype implementation, the routing phases take only about 1-2s, for the test network shown in Fig 7. This is much smaller compared to say, the duration of data collection (40s) or data transfer (Sec. 7).

7. MOBILE DATA TRANSFER

The event detection mechanism triggers data collection at the nodes. After the data *collection* phase, this data must reliably be *transferred* from the (remote) bridge location to a central server that can evaluate the health of the bridge. Most sensor network deployments today do this by first *gathering* all collected data to a sink node in the network. The data is then transferred using some wide-area connectivity technology to a central server. This system rejects this approach for several reasons. First, in a setting where the bridges are spread over a large geographical region (e.g. in Iran), expecting wide area network coverage such as GPRS at the bridge location to transfer data will not be a valid assumption. Setting up other long-distance wireless links, and adds further to maintenance overhead. Having a satellite connection for each bridge is too expensive a proposition. Manual collection of data via period trips to the bridge also means additional maintenance issues. A more important reason is the following. Recall that we can have bridges as long as 2km, with spans of length 30-125m. This means that overall it could have as many as about 200 sensors placed on the bridge, at different spans. Even if we were to have a long-distance Internet link at the bridge location, it would has to *gather* all the sensor data corresponding to these many sensors at a common sink, which has the external connectivity. Such gathering has to be reliable as well. The total data which has to be gathered would be substantial: $57.6Kb \times 200$ _ $1.44MB$ for each measurement cycle. Doing such data gathering over 10-20 hops will involve a huge amount of transfer time and hence considerable energy wastage. Having a large network has other scaling issues as well: scaling of the routing, synchronization, periodic wake-up, command, etc.. Large networks are also more likely to have more fault-tolerance related issues. This then allows us to partition up the entire bridge into several data-spans with *independently operating networks*. In each data-span, the designated head node gathers data from all the nodes within the data-span. It then transfers the data gathered for the data-span onto the train. It then needs to have different trains for data collection and the data transfer. The additional delay introduced in such an approach is immaterial for its application. The transport protocol itself for the data transfer can be quite simple. It uses a block transfer protocol with NACKs for the data transfer from the head node to the train. This system use a similar protocol, implemented in a hop-by-hop fashion, for the data gathering as well: that is, for getting the data from the individual nodes to the head node.

Using multiple channels: It take the approach of using the 16 channels available in 802.15.4. There are at least eight independent channels available. It could simply use different channels on successive spans, and repeat the channels in a cycle. For instance it could use the cycle 1, 3, 5, 7, 9, 11, 13, 15, 2, 4, 6, 8, 10, 12, 14, 16. This would ensure that adjacent channels are at least 7 spans apart, and independent operation of each data-span would be ensured. Note that the train needs to have different motes, operating in the appropriate channel, for collecting data from each data-span.

Reserved channel for event detection: The above approach works except for another subtle detail. It designate that the channel for event detection mechanism for all data-spans is the same, and reserve one channel for this purpose. So as the train approaches, the radio within it can continuously beacon on this reserved channel without having to bother about interfering with any data transfer or other protocol operation within each data span. The head nodes of each span listen on this channel for the oncoming train, and switch to the allocated channel for the data-span after the *Tdet* duration7.

Throughput issues: Another challenge to address in mobile data transfer mechanism is whether such transfer is possible with sufficient throughput. The amount of data that can be transferred is a function of the contact duration of the train with the mote, which in turn is a function of the speed of the train and the antennae in use. In order to determine the amount of data that can be transferred using its hardware platform, it has conducted experiments with a prototype. It needs to transfer blocks of data from the flash of the sender to the flash of the receiver. As mentioned earlier, in the Tmote platform, it cannot perform flash read/write simultaneously with radio send/receive because of a shared bus.

In this implementation, it uses blocks of size 2240 bytes; this is significant chunk of the 10KB RAM. And it has used packets of payload 116 bytes. So a block fits in 20 packets. In this data transfer mechanism, the sender simply uses a *pause* between successive packets to implement flow control. It computes the pause duration as the excess total delay at the receiver side as compared to the sender side. Using an extrapolation of Tab. 2, it calculate the required pause duration for packets of payload 116 bytes (total 126 bytes) to be about 3ms (sender side delay: 8ms, receiver side delay: 11ms). It uses a value of 4ms, as a safety margin. To get an estimate of the various delays involved in the protocol, it first ran a data transfer experiment using an overall file size of 18 blocks. The total time taken was 6,926ms. This consists of the following components.

(1) A flash read-time for each block of about 100ms.

(2) A flash write-time for each block of about 70ms.

This is overlapped with the flash read-time for the next block, except of course for the last block.

(3) The transmission + pause time for each packet is about 12, resulting in an overall transmission + pause time of $20 \times 12 = 240ms$ per block. So it expects a delay of $18 \times 100ms+1\times 70ms+18 \times 240ms = 6190ms$, which is close to the experimentally observed delay of $6, 926ms$.

The above experiment thus gives an effective throughput of $2, 240\times 18 \times 8/6926 = 46.6Kbps$. Note that this value is much lower than the $250Kbps$ allowed by the 802.15.4 radio, due to the various inefficiencies: (a) various header overheads, (b) the shared bus bottleneck mentioned above due to which flash read/write cannot be in parallel with radio operation, and (c) the use of a pause timer to address the flow-control issue.

Mobile data transfer experiment: achieve similar throughput under a realistic scenario, it conducted the following experiment. The setup mimics the situation where the header node (root node) of a cluster in this system uploads the total data collected by all the nodes within its cluster on to mobile node on a train. The data transfer is initiated when the mobile node (on the arriving train) requests the head node to transfer the collected data. The head node then reads the data stored as files from flash and uploads them one by one using the reliable transport protocol. In this experiment, the head node was made to transfer 18 blocks of data, each of 2240 bytes. The total data transfer was thus $18 \times 2240 = 40, 320B$. The antenna of the head node was mounted at a height slightly over 2m. The mobile node was fixed on a vehicle and the antenna was at an effective height of slightly less than 2m. The head node (stationary) with the complete data of 40,320B in its flash is initially idle. The mobile node is turned off until it is taken out of the expected range from the head node, and then turned on. The experiments in Sec. 4.1 show that the range can be around 400m. So it started the mobile node at a distance of 500m from root node. The vehicle is made to attain a constant known velocity at this point of 500m, while coming towards the head node. On booting, the mobile node starts sending the request for data, every 100ms, until it receives a response from the head node. The head node, on receiving the request, uploads the data block by block, using the reliable transport protocol. The transport layer at both nodes takes a log of events like requests, ACKs, NACKs and retransmissions, for later analysis.

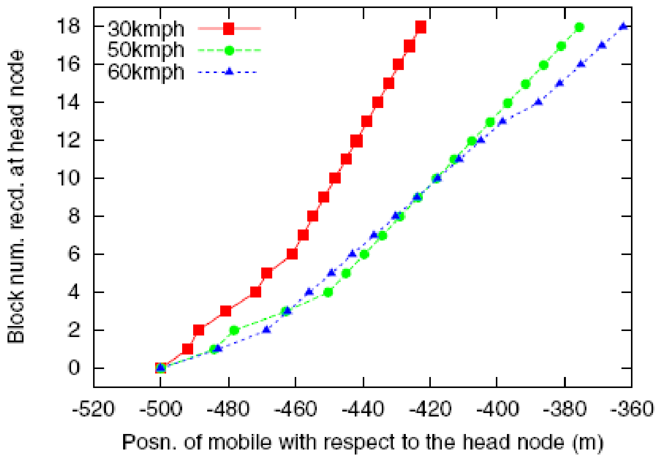


Figure 5: Mobile data transfer measurement

Fig.5 shows a plot of the block sequence number received at the mobile node, versus the position of the mobile with respect to the head node.

8. RELATED WORK

Several projects have looked at the issue of automated structural health monitoring. It focuses on data compression techniques, reliable data transfer protocol, and time synchronization. The work in has looked at bridge monitoring in-depth. They have presented extensive studies of hardware, the required data sampling rate, and data analysis. The novel aspects in our work are the event detection, mobile data transfer, as well as the integration of these aspects with low duty cycling. These have not been considered in earlier work. Low duty cycling by itself is not novel by any means. B-MAC, SCP-MAC and App Sleep are MAC protocols to achieve low duty cycling. Since these protocols have been designed without any *specific* application in mind, they are necessarily generic. Similarly, mobile data transfer too is not novel by itself.

In contrast to the above work on low duty cycling or mobile data transfer, our primary goal is to integrate the requisite functionalities for a specific application. It integrates vertically with *all* aspects relevant to bridge monitoring. It uses *only* the necessary set of mechanisms, thus simplifying the design. It uses extensive application information and cross-layer optimizations across the four

Wider applicability: It has designed specifically for railway bridge monitoring. This environment is particularly challenging since most bridges are away from an urban environment (poor wide area network connectivity). More importantly, train traffic is sporadic and unpredictable. On the other hand, road bridges are likely to have constant and/or predictable (time-of-day related) traffic patterns. However, even in such scenarios, many of these mechanisms are likely to find applicability. For instance, this system uses its event detection mechanism to trigger data transfer, to a designated mobile node. This would be applicable in road bridges too. Also applicable would be the consideration of multiple channels, splitting up the set of nodes into multiple independent networks, and the resulting architecture with the time synchronization and routing functionalities integrated.

Ongoing and future work: Admittedly, many aspects of this work require further consideration. While the prototype implementation and detailed experimentation have given a measure of confidence in this design, actual deployment on a railway bridge is likely to reveal further issues. Related to the issue of fault tolerance, while it has outlined a possible approach to deal with head node (see Sec. 6),

9. CONCLUSION

This paper presents the design of a wireless sensor network based system for long term health monitoring of railway bridges. This is the right way to understand the complex interaction of protocols in this domain. It builds on several aspects of prior work in automated structural monitoring. This novel contributions are three fold: (1) an event detection mechanism which enables low duty cycling, (2) a mobile data transfer mechanism, and (3) the interfacing of these two mechanisms with the time synchronization and routing functionalities. These design choices have been based on application requirements as well as on several measurement studies using prototype implementations. Based on preliminary measurements, it estimates that the current design should be deployable with minimum maintenance requirements: with the battery lasting for over 1.5 years.

10. REFERENCES

1. Kameswari Chebrolu, Bhaskaran Raman, Nilesh Mishra, Phani Kumar Valiveti, Raj Kumar - BriMon: A Sensor Network System for Railway Bridge Monitoring-2008.

2. Otakar ŠVÁBENSKÝ and Pavel ZVĚŘINA, Czech Republic - Deformation Measurement of Railway Bridge Abutment Pier - INGEO 2004 and FIG Regional Central and Eastern European Conference on Engineering Surveying Bratislava, Slovakia, November 11-13, 2004

3. Daniele Inaudi, Nicoletta Casanova, Pascal Kronenberg, Samuel Vurpillot - Railway Bridge monitoring during construction and sliding- SPIE Conference on Smart Structures and Materials, 5-6.03.1997, San Diego, USA