Chapter 2

Activity Scheduling in Bluetooth Sensor Networks

Jelena Mišić, G.R. Reddy, and Vojislav B. Mišić

Contents

2.1 Overview

Sensor networks are finding widespread use in diverse application areas such as environmental monitoring, health care, logistics, surveillance, and others. In most cases, the sensor network must satisfy the two simultaneous, yet often conflicting goals of maintaining a desired packet arrival rate at the network sink and maximizing the network lifetime. The problem is further compounded by the fact that most of those networks operate on battery power with as little human intervention as possible. This chapter considers a Bluetooth scatternet operating as the sensor network with a medium data rate sensing application. In this setup, we present an approach to activity management in Bluetooth sensor networks that utilizes cross-layer adaptive sleep management on a per-piconet basis. The effects of finite buffers in individual nodes is also considered. The proposed approach is shown to be computationally simple yet effective.

2.2 Introduction

A wireless sensor network consists of a number of wireless sensor nodes, spread across a given area, that are used for event detection and reporting [1]. In the sensor network, a node (sink) issues queries that request data from the sensing node; a group of such nodes that can provide the requested data (known as the source) sends it to the sink [3]. Because sensor networks are primarily used for event detection tasks, the rate at which data is propagated from source node(s) to the sink must be high enough to obtain the desired reliability R, which is commonly defined as the number of data packets required per second for reliable event detection at the sink [2]. At the same time, sensor networks frequently operate on battery power, which means that energy efficiency must be maintained.

Reliable event detection using minimal energy resources requires simultaneous achievement of several sub-goals. The packet loss along the path from the source to the sink must be minimized. At the PHY layer, packets can be lost due to noise and interference, while at the MAC layer, losses may be incurred by collisions. Because sensors are continuously monitoring the environment and sending data, retransmission of lost packets is not necessary; it would use up the bandwidth to send stale data, and thus impair the performance of the sensor network in qualitative terms.

Packet delays must be minimized as well, including queueing delays experienced in various devices along the data path and also delays due to congestion in the network. (Queueing delays are incurred at the MAC layer, while congestion detection and control are performed at the transport layer.)

Finally, packet propagation should take place along the shortest paths, while avoiding congested nodes and paths; this is the responsibility of the network layer.

The sensor nodes are generally battery operated and have limited computational capabilities, which in turn means that the protocol stack must be as simple as possible. As a result, the simultaneous goals of minimizing packet losses and maximizing the efficiency (and, by extension, maximizing the lifetime of the network as well) necessitate that some of the aforementioned functions of the different layers are performed together. That is, cross-layer optimization of network protocol operation is needed; the feasibility of this optimization is determined by the communication technology used to implement the network.

This chapter describes a Bluetooth sensor network operating under the scheme that integrates congestion control with reliability and energy management. We investigate the performance of the proposed solution and the ways in which its operation can be optimized. We start by discussing the suitability of Bluetooth technology for use in sensor networks, and present the most important among the existing solutions for congestion control and energy management problems in sensor networks. Then we develop two algorithms that schedule the sleep of individual slaves: the first one maintains fixed event reliability at the sink and the second one keeps the satisfactory event reliability by avoiding congestion. The performance of the proposed algorithms is analyzed in detail. A note on our simulation setup and a brief summary conclude the chapter.

2.3 Bluetooth and Sensor Networks

Bluetooth was originally intended as a simple communication technology for cable replacement [6]. However, its use has been steadily growing in a diverse set of applications [2]. Bluetooth operates in the Industrial, Scientific and Medical band at 2.4 GHz using the Frequency Hopping Spread Spectrum technique, which makes it highly resilient to the noise and interference from other networks operating in the same band, such as IEEE 802.11b and IEEE 802.15.4 [8]. The raw data rate of 1 Mbps (or 3 Mbps, if the recent version 2.0 of the standard is used) and the default transmission range of 10 to 100 meters [6] make Bluetooth networks suitable for medium-rate wireless personal area networks (WPANs). These same qualities mean that Bluetooth is suitable for the construction of low-cost sensor networks, offering coverage of sensing areas with diameters of several tens to several hundred meters [2].

2.3.1 Piconet Operation

Bluetooth devices are organized into piconets, small networks with up to eight active nodes and up to 255 inactive ones [6]. Bluetooth uses a

TDMA/TDD polling protocol where all communications are performed under the control of the piconet master. Bluetooth uses a set of RF frequencies (79 or 23, in some countries) in the ISM band at about 2.4 GHz. The Frequency Hopping Spread Spectrum (FHSS) technique is utilized to combat interference. Each piconet hops through the available RF frequencies in a pseudo-random manner. The hopping sequence, which is determined from the Bluetooth device address of the piconet master, is known as the channel [6]. Each channel is divided into time slots of $T = 625\mu$s, which are synchronized to the clock of the piconet master. In each time slot, a different frequency is used.

All communications in the piconet take place under the control of the piconet master. All slaves listen to downlink transmissions from the master. The slave can reply with an uplink transmission if and only if addressed explicitly by the master, and only immediately after being addressed by the master. Data is transmitted in packets, which take one, three, or five slots; link management packets also take one slot each. The RF frequency does not change during the transmission of the packet. However, once the packet is sent, the transmission in the next time slot uses the next frequency from the original hopping sequence (i.e., the two or four frequencies from the original sequence are simply skipped). By default, all master transmissions start in even-numbered slots, while all slave transmissions start in odd-numbered slots. A downlink packet and the subsequent uplink packet are commonly referred to as a frame. Therefore, the master and the addressed slave use the same communication channel, albeit not at the same time. This communication mechanism, known as Time Division Duplex, or TDD for short, is schematically shown in Figure 2.1. This approach is collision-free and, consequently, more energy efficient than the collision-based MACs used in IEEE 802.11 and IEEE 802.15.4 [11].

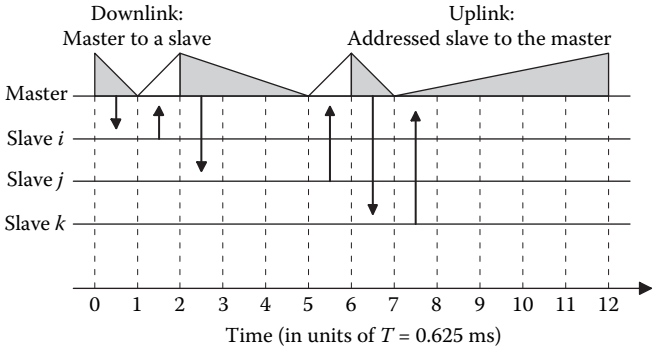


Figure 2.1 TDD master-slave communication in Bluetooth. Gray triangles denote data packets; white triangles denote empty (POLL and NULL) packets.

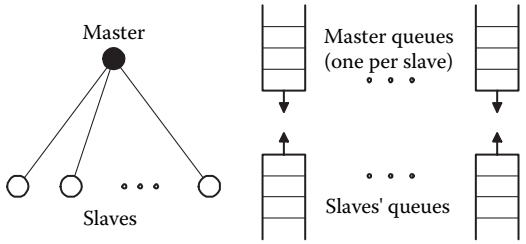


Figure 2.2 The Bluetooth piconet and its queueing model.

Because of the TDD communication mechanism, all communications in the piconet must be routed through the master. Each slave will maintain (operate) a queue where the packets to be sent out are stored. The master, on the other hand, operates several such queues, one for each active slave in the piconet. The piconet and the corresponding queueing model is shown in Figure 2.2. We note that these queues may not physically exist, for example, all downlink packets might be stored in a single queue; however, the queueing model provides a convenient modeling framework that facilitates the performance analysis of Bluetooth networks.

2.3.2 Intra-Piconet Polling

The master polls the slave by sending the data packet from the head of the corresponding downlink queue. The slave responds by sending the data packet from the head of its uplink queue. When there is no data packet to be sent, single-slot packets with zero payload are sent — POLL packets in the downlink and NULL packets in the uplink direction [6]. As the process of polling the slaves is actually embedded in the data transmission mechanism, we use the term "polling" for every downlink transmission from the master to a slave.

Because packets must wait at the slave or at the master before they can be delivered to their destinations, the delays they experience are mainly queueing delays. Therefore, the intra-piconet polling scheme is obviously the main determinant of performance of Bluetooth piconets, and one of the main determinants of performance of Bluetooth scatternets. As usual, the main performance indicator is the end-to-end packet delay, with lower delays being considered as better performance. There are, however, at least two other requirements to satisfy. First, the piconet master should try to maintain fairness among the slaves, so that all slaves in the piconet receive equal attention in some shorter or longer time frame. (Of course, their traffic load should be taken into account.) Second, Bluetooth devices are, by

default, low-power devices, and the polling scheme should be sufficiently simple in terms of computational and memory requirements.

The polling schemes can roughly be classified according to the following criteria:

- The number of frames exchanged during a single visit to the slave can differ; it can be set beforehand to a fixed value, or it can be dynamically adjusted on the basis of current and historical traffic information.
- Different slaves can receive different portions of the bandwidth; again, the allocation can be done beforehand, or it can be dynamically adapted to varying traffic conditions. The latter approach is probably preferable in Bluetooth piconets, which are ad hoc networks formed by mobile users, and the traffic can exhibit considerable variability. In fact, due to users' mobility, even the topology of the piconet can change on short notice. However, the fairness of polling might be more difficult to maintain under dynamic bandwidth allocation.
- Finally, the sequence in which slaves are visited can be set beforehand, or it can change from one piconet cycle to another, depending on the traffic information. In either case, slaves that had no traffic in the previous cycle(s) can be skipped for one or more cycles, but the polling scheme must ensure that the fairness is maintained.

The current Bluetooth specification does not specifically require or prescribe any specific polling scheme [6]. This may not seem to be too big a problem, because optimal polling schemes for a number of similar single-server, multiple-input queueing systems are well known [17,18]. However, the communication mechanisms used in Bluetooth are rather specific and the existing results cannot be applied. It should come as no surprise, then, that a number of polling schemes have been proposed and analyzed [7,13]. Many of the proposed schemes are simply variations of the well-known limited and exhaustive service scheduling [24], but several improved adaptive schemes have been described as well [13,15].

In our work, we have chosen the so-called *E-limited service* polling scheme in which the master stays with a slave for a fixed number M of frames ($M > 1$), or until there are no more packets to exchange, whichever comes first. Packets that arrive during the visit are allowed to enter the uplink queue at the slave and can be serviced — provided the limit of M frames is not exceeded [24]. This scheme has been found to offer better performance than either limited or exhaustive service, and the value of M can be chosen to achieve minimum delays for given traffic burstiness [20].

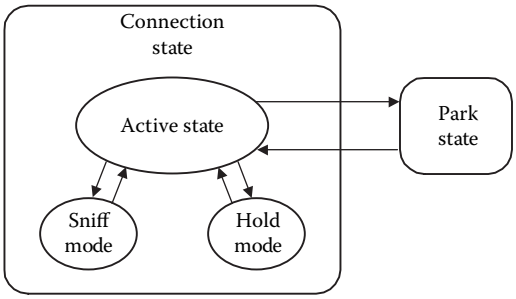


Figure 2.3 Connection states and modes.

2.3.3 Power-Saving Operation Modes

An active slave is assigned a three-bit active mode address AM_ADDR that is unique within the piconet [6]. However, the slave need not listen to master transmissions all the time; it may choose to switch to one of the so-called modes in which it can detach itself from the piconet for prolonged periods without having to surrender its piconet address; these modes are known as HOLD and SNIFF. The slave can also switch to a parked state, in which it releases its active mode address; this switch can be initiated by either the master or the slave itself. Broadcast as well as unicast messages can target active or parked slaves only, as appropriate. The connection states and modes are schematically shown in Figure 2.3.

An active slave can temporarily detach itself from the piconet by entering the so-called HOLD mode, the operation of which is shown in Figure 2.4. In this mode, the master will not poll the slave for a specified time interval, referred to as the *holdTO*, or hold timeout. The inactivity period due to the HOLD mode affects only ACL links established between the master and the slave; SCO and eSCO links, if any, remain operational even when the slave is in the HOLD mode. During the HOLD mode, the slave can engage in other activities such as scanning, paging, or joining another piconet. The slave can also enter a low power mode to conserve energy.

The actual duration of the HOLD mode is negotiated between the master and the slave; the negotiation process can be initiated by either the master or the slave itself. The initiating party proposes the switch to the HOLD mode as well as the hold timeout; the responding party can accept it, or respond with a counterproposal of its own.

Another mode that can be entered from the active connection state is the SNIFF mode, the operation of which is shown in Figure 2.5. In this mode, the slave is absent from the piconet for a specified time, during which the master will not poll it. The slave periodically joins the piconet to listen to master transmissions. If no transmission is initiated, or even detected,

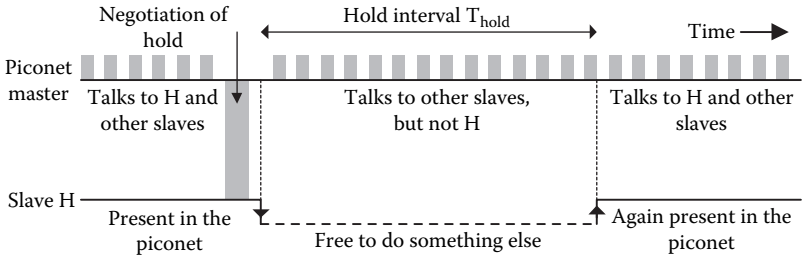


Figure 2.4 The operation of the HOLD mode.

during the predefined window, the slave again detaches itself from the piconet for another interval of absence. During this time interval, the slave can engage in other activities, similar to the HOLD mode described above. Again, the inactivity due to SNIFF mode affects only the ACL link or links that may be set between the master and the slave in question, but not the SCO or eSCO ones, if any.

As is the case with the HOLD mode, the SNIFF mode and its parameters are negotiated between the master and the slave. The negotiation process can be initiated by either party; the initiating Link Manager (LM) proposes the SNIFF mode and its parameters to the corresponding LM of the other participant. Once the switch and the parameters are accepted, the slave can start alternating between active and SNIFF mode. Unlike the HOLD mode, which is a one-off event, the SNIFF mode lasts until one of the participants explicitly requests its termination.

A connection can break down for different reasons, including power failure, user movement, or severe interference. To detect the loss of connection, the traffic on each link must be monitored on both the master and the slave side. This is accomplished through the so-called supervision

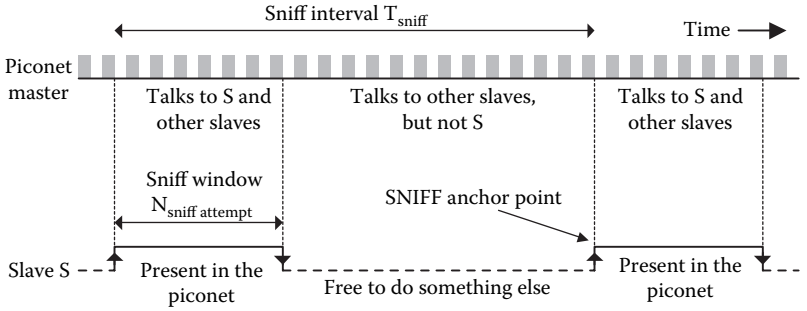


Figure 2.5 The operation of the SNIFF mode.

timer, $T_{supervision}$, which is reset to zero every time a valid packet is received on the associated physical link. If the timer reaches the *supervisionTO* timeout, the value of which is negotiated by the Link Manager, the link is considered lost, and the associated active piconet member address can be reassigned to another device. The value of the supervision timeout should be longer than negotiated HOLD and SNIFF periods. The same link supervision timer is used for all logical transports carried over the same physical link.

2.3.4 Bluetooth Scatternets

A group of independent piconets interconnected through shared devices, or bridges, forms a scatternet; this work focuses on slave-slave bridges that act as slaves in each of the piconets they visit. As most Bluetooth devices — bridges included — have only one radio interface, the bridge must visit adjacent piconets in different time periods. Consequently, both the intra-piconet polling scheme and the inter-piconet (bridge) scheduling scheme are important factors that determine the performance of a Bluetooth scatternet [21].

Most of the schemes are based on the concept of rendezvous points: time instants at which the bridge should be present in the piconet to exchange data with its master [14]. These time instants can be fixed before the actual data transfer, maybe even for the entire lifetime of the scatternet, or they can be negotiated as necessary between the piconet master(s) and the bridge(s). The rationale for the existence of a predefined rendezvous point is to have both participants join the exchange simultaneously. If this is not the case, the participant that switches earlier would have to wait idle and thus waste time and, ultimately, bandwidth.

The schedule of rendezvous points can be fixed beforehand or adaptive. The former case may be suitable for sensor networks that have comparatively well-known traffic requirements; it certainly is unsuitable for sensor networks that feature activity management in which sensors are going to sleep for prolonged periods of time. This case seems easier to handle using the latter approach with adaptive scheduling of rendezvous points. But in either case, the main problem with rendezvous-based bridge scheduling remains: the overhead incurred by the construction and maintenance of the schedule of rendezvous points.

This overhead can be avoided if the bridge (or bridges) could operate without such a schedule. It turns out that such an approach, which will be referred to as *walk-in bridge scheduling*, is indeed feasible [22]. Under walk-in scheduling, the bridges can switch between piconets at will, without any prior arrangement. The piconet masters will poll their slaves as determined by the chosen intra-piconet polling scheme, which includes the bridge as well as other slaves. The master will, therefore, poll the bridge in

each piconet cycle, and the exchange will start only if the bridge is found to be present.

The main advantage of the walk-in scheme lies in the absence of rendezvous points, which means that any given piconet can accommodate several bridge devices simultaneously, and any given bridge can visit several piconets in sequence. Walk-in bridge scheduling can thus be applied with ease in scatternets of arbitrary size, and there is no performance penalty due to the construction and subsequent maintenance of the schedule of rendezvous points. Neither of these features can be achieved with rendezvous-based scheduling.

2.4 Related Work

A number of schemes have been proposed for event detection and data transmission in wireless sensor networks, most notably the following.

Directed diffusion (DD) has been proposed for event detection and reliable data transfer in wireless sensor networks [12]. In this scheme, a node requests data by sending an interest query for named data; once a match for the required data is found, the results are transferred to the querying node. In this process, intermediate nodes can aggregate the obtained data, store them in their cache, and redirect them to their neighboring nodes. However, to store the interest queries and the resulting data sets, the DD scheme assumes that all nodes are roughly equivalent in terms of computational and memory capabilities. This might cause significant overhead for sensor networks, which generally have serious power and processing limitations [9]. Furthermore, the guaranteed end-to-end data delivery (which DD supports) is not required for event detection, due to the fact that correlated data flows from several source nodes are loss tolerant as long as event features are reliably detected [2,4].

Pump Slowly Fetch Quickly (PSFQ) scheme is based on propagation of data from the source node by injecting data at relatively low speed and allowing nodes that experience data loss to fetch any missing data packets from immediate neighbors by requesting retransmission [26]. The main source of packet loss in this scheme is the poor quality of wireless links and the resulting transmission errors, while traffic congestion and resulting packet blocking due to buffer overflows at various stages in the network are not considered [2]. This is not a realistic assumption for sensor networks, especially in view of the fact that some packet loss may be acceptable due to correlation of sensed data.

Event-to-Sink Reliable Transport (ESRT) has been designed for reliable event detection [2]. ESRT utilizes the notion of event-to-sink (multipoint-to-point) reliability, rather than the more common end-to-end (point-to-point) reliable transport protocols. The philosophy of ESRT is to prevent

the nodes from sending extra packets but, at the same time, not to optimize the number of nodes required to sense the data for a given task. Also, ESRT transfers raw data to the sink without any kind of in-network processing [10,25].

In the energy-efficient **CODA transport protocol** [25], the proposed congestion handling technique uses heuristic mechanisms for monitoring network operations to avoid congestion. These mechanisms include receiver-based congestion detection, open-loop hop-by-hop back-pressure, and closed-loop multi-source regulation. In receiver-based congestion detection, CODA uses combinations of present and past channel loading conditions and current buffer levels to predict congestion occurrence in the network. A simple technique is used for monitoring the message queue. Although CODA achieves congestion control, the messaging overhead required in controlling congestion leads to higher energy consumption. Also, congestion control is not directly connected to the application reliability at the sink.

The aforementioned approaches can be roughly classified into those that achieve individual packet reliability using packet retransmission, and those that try to obtain a sufficient number of packets at the sink using some kind of feedback to inform the sensing nodes to decrease the reporting rate. Neither of them considers the effects of finite buffer limitations of sensing and bridging devices.

Furthermore, all these approaches either do not consider the impact of the MAC protocol at all or assume the use of collision-based (and thus generally inefficient) protocols such as CSMA-CA. This was the motivation that led us to investigate the possibility of implementing wireless sensor networks using Bluetooth and its collision-free MAC protocol.

The suitability of Bluetooth as the platform to implement sensor networks has been investigated by building a Bluetooth protocol stack for the TinyOS operating system [16]. The experiments were conducted on actual Bluetooth devices, known as BTnodes, which were developed at ETH Zurich [5]. The BTnodes were equipped with two radios to enable multihop networking. The network was then tested for throughput and energy consumption. The results suggest that Bluetooth-based sensor networks could be appropriate for event-driven applications that exchange bursts of data for a limited time period.

One possible limitation for the use of Bluetooth to implement sensor networks is the limited number of slaves. As mentioned above, a Bluetooth piconet can have, at most, seven active slaves at any given time, while up to 255 others can be parked [6]. Consequently, sensor networks with a large number of nodes must be implemented either as Bluetooth scatternets, or perhaps by combining Bluetooth with other communication technologies such as IEEE 802.15.4 [11]. This is a promising area for further research.

2.5 Congestion Control in Sensing Scatternets

Let us consider a Bluetooth sensor network implemented as a scatternet such as the one shown in Figure 2.6. (In an earlier work, that same topology was used to assess the impact of finite buffer size on scatternet performance [21].) In this setup, one of the nodes in piconet P_1 acts as a sink, while all other nodes act as sources. Each piconet represents a cluster of sensor nodes that is controlled and coordinated by the piconet master. Each slave maintains an uplink queue toward the master, and for each slave, the master maintains a downlink queue. The master also maintains outgoing downlink queues for each bridge. Each bridge has one incoming queue and several outgoing queues, one per each piconet it visits. When the sink needs to acquire some information from the network, it injects a query that is propagated through the network. Once the query reaches the source nodes, they take actions to respond to the query: they collect the data and send it back to the sink.

The traffic model depends on the sensing application. Consider a relatively high-bandwidth, low-cost, surveillance-based sensing application

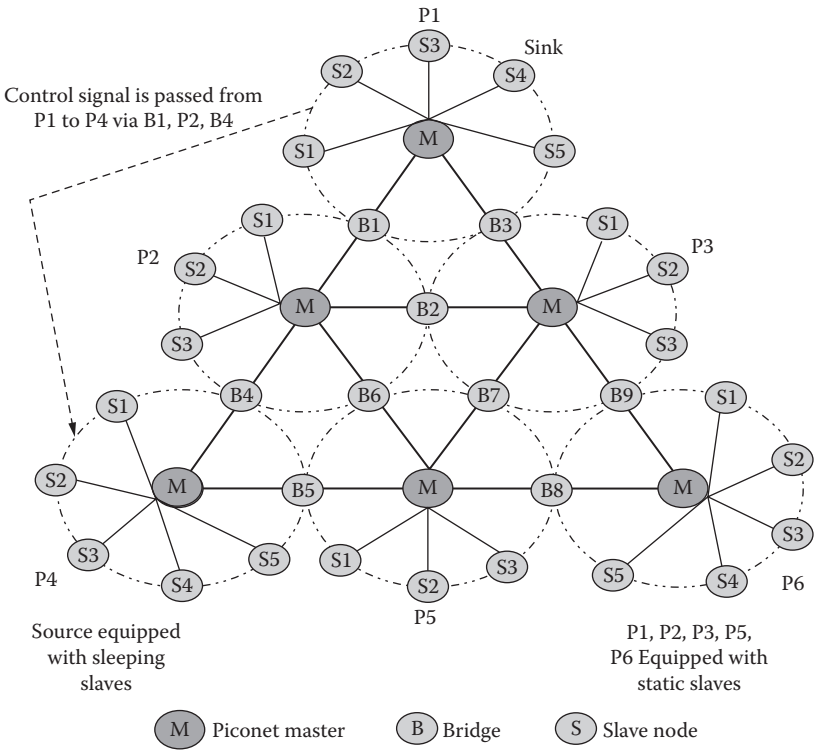


Figure 2.6 Wireless sensor network with triangular topology.

where compressed still images are taken as a result of event detection and sent to the sink. This setup can be used in applications such as road traffic control or asset protection. It is unlikely that the sensed data will fit in a single Bluetooth packet — even with the Enhanced Data Rate facility defined in Bluetooth v2.0, the largest packet size is still only 1023 bytes [6] — and, therefore, the traffic must consist of bursts of several packets. We assume that the packet burst size is geometrically distributed with mean burst size $\overline{B}$ (in this case, $\overline{B} = 3$), and each packet has a length of five Bluetooth time slots T. The packet burst arrival rate λ to each sensing node is presented as the number of packet burst arrivals per time slot. For example, values used in our model are 0.002 to 0.005 packet bursts (images) per time slot, which translates into approximately 3.2 to 8 packet bursts (images) per second. The packet burst arrival rate and the probability of traffic locality are uniform for all the slaves.

Furthermore, we assume that the nodes within a piconet can exchange some other data with the master as required by the sleep management algorithm. Therefore, the locality probability (i.e., the probability that the traffic generated by the slave will have destinations in the same piconet) is set to some small value P_l; the complementary probability that the destination is in another piconet (sink) is $1 - P_l$. When the traffic is generated by a slave for an other piconet, the packets are routed through the intermediate piconets, via bridges and piconet master(s), to the destination piconet by taking the shortest path. For simplicity, we assume that neither the masters nor the bridges generate any traffic. We also assume that intra-piconet polling uses the E-limited scheduling scheme, while bridge scheduling is performed using the walk-in approach [19].

Considering event reliability, we can distinguish between no less than three related, yet quite distinct concepts:

- Absolute event reliability corresponds to the number of packets received per second at the sink from all source piconets. We can also introduce absolute event reliability per source piconet, which is the number of packets per second received by the sink from the given source piconets.
- Relative reliability is defined as the ratio of the number of received packets from the source piconet at the sink and the number of transmitted packets by that piconet.
- Finally, the desired reliability is the number of data packets required for reliable event detection at the sink. This number is determined by the requirements of the sensing application.

Our initial exploration focused on the relative reliability per source piconet. When the packet arrival rate (and, consequently, the traffic load)

increases, the reliability will increase but only up to a certain point, which can be explained by the lack of congestion control. Namely, the increase in traffic toward the sink will, at some point, overload the bridge buffers along the way. As a result, bridge buffers will begin to drop packets, which leads to a reduction in relative reliability for traffic from a given piconet.

One approach to minimizing packet losses at the bridge buffers and maximizing relative reliability at the sink would be to control the traffic load; this can be accomplished by controlling the number of active slaves in all the source piconets. Because the operation of a piconet is entirely controlled by the piconet master, it is the piconet master that needs to instruct slaves to temporarily suspend their activities; this is performed at the request of the network sink. (Should reliability fall below a predefined limit, the sink can request the master to increase the number of active slaves.) Activation and deactivation can be accomplished by unparking some parked slaves and parking previously active slaves.

An alternative (and much faster) procedure is to put active slaves in one of the possible power-saving modes, such as SNIFF or HOLD [6]. In both of these, the slave in question retains its network address, although the master will not try to poll it. In our experiments, we have assumed that the slave will enter a low-power mode and thus conserve energy. Upon returning to active state, the slave again begins to listen to the master's transmissions, while the master is free to poll the slave at will.

While both SNIFF and HOLD modes could, in theory, be utilized to implement the power-saving mechanism, the HOLD mode has a distinct advantage. Namely, the duration of each HOLD interval is negotiated anew between the master and the slave in question, which opens the possibility for adjustment to any desired time interval. The SNIFF mode, on the other hand, entails distinct procedures for initiation and termination, which makes it less suitable for our purposes. Overall, the use of the HOLD mode gives us both the effectiveness and flexibility of the procedure, which is why we have chosen to implement the activation and deactivation of slaves using the HOLD mode.

2.6 Maintaining Fixed Reliability at the Sink

Let us assume that N_p piconets are reporting the sensing information to the master in the sink piconet. Piconets are indexed by index $i = 1 \ldots N_p$, and each piconet P_i has m_i ordinary slaves (i.e., slaves without the bridging function). The upper limit of reliability of sensed information is determined by the piconet capacity. If five slot packets are used and there is no downlink data traffic (which, in fact, is needed to carry control information), the maximum absolute reliability is $R_{max} = 1/(T + 5T) = 266$ packets per second, where $T = 625\,\mu$s is the duration of Bluetooth time slot.

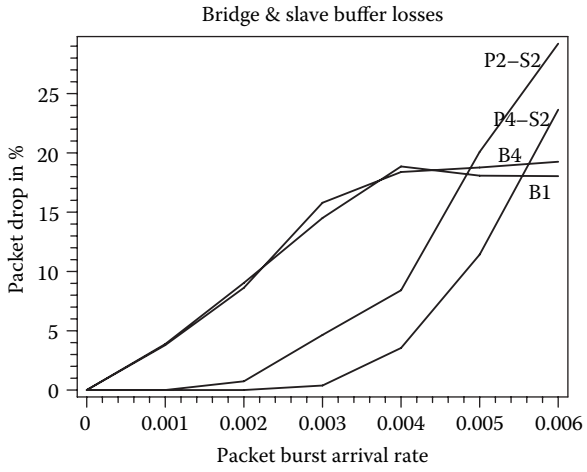


Figure 2.7 Blocking probability versus offered load at the slaves in P_2 and P_4 and the bridges B_1 and B_4.

In practice, the maximum achievable number will be lower, due to the losses at the bridges and presence of downlink traffic needed to send queries and control information. In many cases it will suffice to maintain the reliability at some application-defined level R; assuming uniform conditions, the absolute reliability contributed by each piconet is $R_i = R/N_p$.

We also need to estimate packet losses at the slave and bridge buffers. The bridge loss rate is a function of total piconet load, bridge load, bridge polling parameter M_b, slave polling parameter M_s, and bridge buffer size. In case the topology is fixed and the polling parameters are known, we can assume that the bridge loss rate depends on the bridge packet arrival rate and total piconet load. In this case, the bridge loss rate can be approximated with $P_{b,i} = K_i \overline{B} L \lambda_{b,i}$, where $\overline{B}$ is the average burst size, L is the packet size in slots, $\lambda_{b,i}$ is the burst arrival rate toward the bridge, and K_i is the proportionality constant [19]. Measured values of blocking probabilities are shown in Figure 2.7.

Therefore, the sink can calculate losses from source piconets and communicate them to source piconets to adjust the slaves' activities. Of course, these losses should not be too high — say, up to a few percent — otherwise, the network is operating in the congestion regime, in which case it is better to partition it into sections with separate (and different) sinks and thus avoid congestion. When losses along the path are known, the source piconet can compensate for the losses by scaling its absolute reliability to $R'_i = \frac{R_i}{\prod_{over\ path}(1-P_{b,i})}$. The absolute reliability must be transformed into the average number of active slaves per piconet. The mean number of packets

contributed by the slave per second is $R_s = \lambda \overline{B}/T$, while the mean number of active slaves per piconet is $A_{s,i} = \frac{R_i' T}{\lambda \overline{B}}$.

Procedure LONG : managing the long activity period

Data: a, b, m_i
Result: initial value of the short activity management counter C_s
begin
  if $a \leq b$ **then**
    put $m_i - 1$ most recently used slaves to HOLD mode for bT_u seconds;
    remaining slave should be active for aT_u seconds;
  else
    put $m_i - \lceil \frac{a}{b} \rceil$ most recently active slaves to sleep for bT_u;
    among $\lceil \frac{a}{b} \rceil$ remaining slaves, activate $\lfloor \frac{a}{b} \rfloor$ least recently used slaves for next bT_u seconds;
    the remaining slave S^* should be active for $(a \bmod b)T_u$ seconds;
  end
  $C_s = a \bmod b$,
end

The mean value of $A_{s,i}$ slaves at any given time can be obtained in the following manner. Assume that $A_{s,i}$ is a rational number: $A_{s,i} = \frac{a}{b}$, where a, b are integers. Further assume that the activity control process consists of basic time units T_u when the slave can be put in HOLD state. (Note that T_u should be much larger than the Bluetooth time slot; in this work, we assume that T_u is one second.) Then, a units of activity must be executed by the slaves over every b time units. The values for a and b should be selected according to the desired level of granularity of the sleep control. Let us denote the long activity management period with bT_u, and the short activity management period with T_u.

Procedure SHORT : managing the long activity period

Data: C_s
begin
  if $C_s > 0$ **then**
    $C_s = C_s - 1$;
  else if $C_s = 0$ **then**
    $C_s = C_s - 1$;
    put slave S^* to HOLD for $(b - a \bmod b)T_u$ seconds;
end

Within the long activity management period, we try to minimize the number of slaves needed to accomplish this activity requirement. In effect,

this is an attempt to minimize the protocol overhead because the slaves will sleep in the HOLD mode and this must be negotiated; the less negotiation we undertake, the more efficient the protocol becomes.

During the short management cycles, we will try to balance the utilization of various slaves in an effort to extend the battery life of each slave. Additionally, feedback from the sink can be communicated to the source piconets to slightly decrease or increase the average number of active slaves, which will result in decrementing or incrementing the value of a.

In this manner, we are able to maintain the reliability at the sink at the desired level. The entire procedure is shown in Algorithm 1.

Algorithm 1: Maintaining fixed reliability at the sink

```
Data: total event reliability at the sink R, scatternet topology, N_p, m_i,
      i = 1...N_p, packet burst arrival rate λ per slave, mean burst size B̄
begin
    for each piconet P_i do
        estimate event reliability R_i;
        estimate load through outgoing bridges;
        estimate packet loss through each bridge;
        estimate total packet loss towards the sink;
        recalculate R'_i;
        find A_{s,i}, a, b;
        C_0 = 0;
        after every T_u seconds do
            C_0 = C_+1;
            management of long activity period;
            if (C_0 mod b == 0) then
            |  call LONG;
            end
            management of short activity period;
            call SHORT;
        end
    end
end
```

To validate this algorithm, we performed simulation experiments with the required event reliability of 20 packets per second at the sink from all the piconets. The packet burst arrival rate for each slave, when active, was set to $\lambda = 0.001$. The reliability requirement was mapped into bridge packet burst arrival rates, and losses through the bridges were estimated as 3 percent for B_4 and 5 percent for B_1, respectively. Then the source piconet transmission rates were set to 4.3 packets per second for P_4, P_5, and P_6 and 4.21 packets per second from P_2 and P_3. The resulting activity of the

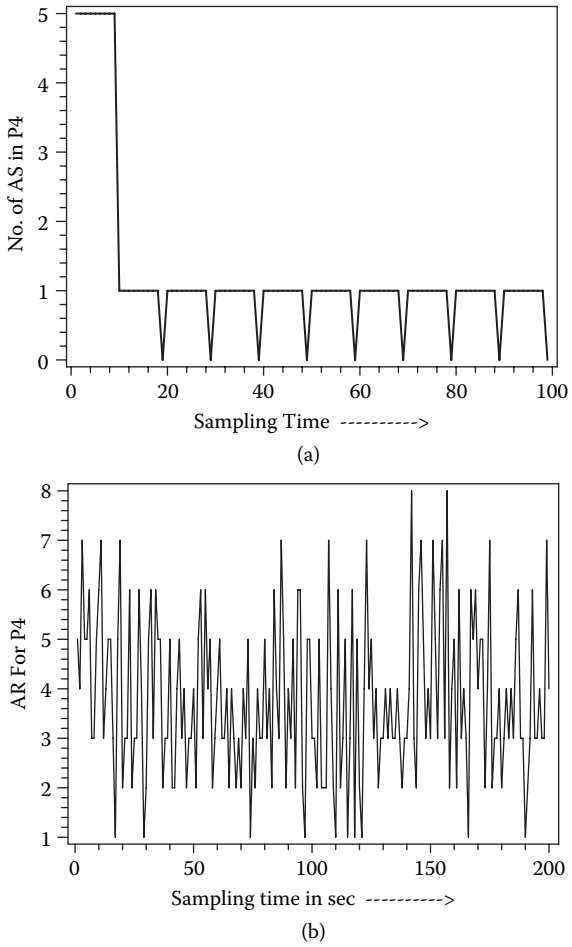


Figure 2.8 Mean number of active slaves and absolute reliability at the sink, for packets from slaves in P_4: (a) number of active slaves over time, and (b) absolute reliability from P_4 at the sink.

slaves in piconet P_4 and the reliability at the sink are shown in Figure 2.8; as can be seen, the algorithm manages to maintain the mean value of absolute reliability around the desired value, while the number of active slaves is minimized.

2.7 Optimizing Reliability at the Sink

While the algorithm described above manages to maintain the reliability at the desired level, it does so without respect for other considerations, in particular the congestion level and the losses due to finite buffers at

the bridges and masters through which the packets must pass. Fortunately, a scheme can be devised to simultaneously perform sleep management *and* congestion control, and thus accomplish reliable event detection while minimizing energy consumption.

In some cases, it might be desirable to operate the network in the area of no congestion or mild congestion so as to have minimal losses and to extend the lifetime of the network. (This may be likely to happen when the process being observed changes very slowly over time.) To that end, let us define the relative reliability of the event from a piconet P_i as the ratio of the number of packets generated by the piconet and the number of those packets that are actually received by the sink: $RR_i = \prod_{overpath} (1 - P_{b,i})$. Relative reliability depends on the network load and can be used to detect congestion.

Using the scatternet topology from Figure 2.6 as an illustration, the algorithm operates as follows. Initially, the exterior piconets operate with five active slaves, while the interior ones operate with only three, because of their higher carried load. The desired reliability is chosen by the user; the actual reliability is periodically calculated at the sink and communicated to the source piconet (i.e., to its master, which then manages individual slave activity).

Then, the piconet master is able to calculate the relative reliability over the period that is a multiple of the long activity management period. Given that the length of the long management period is bT_u, the length of period for estimating reliability is cbT_u. We have chosen $b = 10$, $T_u = 1$ s, and $c = 6$; those values give 60 s as the period for estimating reliability and 10 s for the period for changing total slave activity. As in the example with fixed reliability, the total slave activity is calculated as a rational number $A_{s,i} = a/b$, where $b = 10$ for simplicity. Slave activity (expressed through the variable a, the current value of which is denoted as a_c) can be changed only at the boundaries of reliability estimation period. The algorithm also maintains the history of slave activity, as an exponentially weighted moving average a_b; it increases slave activity only if there is an increasing trend of activity and the relative reliability is below the threshold. (In our case, the smoothing constant is $\alpha = 0.5$.)

The number k regulates the step of algorithm progression. It can be set to any value between 1 and b that corresponds to exclusion of one slave. Higher values of k result in faster reactions with possible oscillations, while smaller values lead to slow adaptation to network conditions. We found that $k = b$ gives satisfactory behavior of the algorithm.

When the network experiences congestion, some of the data packets transferred from the source area to the sink are lost due to buffer overflow at the intermediate bridges. This overflow results in a sudden drop in the relative event reliability sensed by piconets, which is taken as a sign of congestion and low reliability in the network. At this point of time, the

event reliability calculated at the source master (based on the measurement taken at the sink) will drop below the desired event reliability. Hence, the source master must increase the duration of the hold mode for its slaves. Once a slave is put into HOLD mode, it stops sending the sensed data to the source master, thereby conserving its battery power and reducing congestion in the network. Once the sleep time expires, the slave returns from the HOLD mode and starts collecting data and is ready to be polled.

Au: There is no text mention for Algorithm 2. Please add.

Algorithm 2: Controlling relative reliability at the sink (algorithm is executed at each piconet)

```
Data: piconet index i, m_i, penalty k, measured reliability R_i
begin
    while true do
        after every cbT_u seconds
            C_0 = C_0 + 1
            if (C_0 mod b == 0) then
                call LONG;
            call SHORT;
        end
        receive measured reliability from the sink;
        calculate RR_i, a_c = a;
        if RR_i > T_H then
            a = a_c − k;
        else if (RR_i < T_L) AND (a_b > a_c) then
            a = a_c + 1;
        else if (RR_i < T_L) AND (a_b < a_c then
            a = a_c − k;
        end
        a_b = αa_b + (1 − α)a_c
    end
end
```

Table 2.1 shows a representative measured trace from the simulator to illustrate our sleep regulation technique. In this case, sleep management is applied to all piconets.

We note that the packet transmission rates will be affected by the packet loss caused by noise and interference. While this packet loss is indeed possible, its effects will be negligible because of the following:

- Bluetooth uses Frequency-Hopping Spread Spectrum (FHSS), which makes it rather resilient to noise and interference [27].
- Bluetooth packets can be protected using Forward Error Correction (FEC), at the expense of slight reduction of their information carrying capacity.

Table 2.1 Simulator Trace for P_4

Interval	*Relative Reliability*	*Active slaves in P_4*	*Slaves*	
			Put on HOLD	*Back from HOLD*
1	55	5	0	0
2	64	5	2	0
3	75	3	2	0
4	88	1	0	0
5	86	2	1	1
6	85	3	2	2
7	90	3	2	2
8	91	2	1	1
9	88	3	2	2
10	89	3	2	2
11	91	2	1	1
12	88	3	2	2
13	92	3	2	0
14	88	1	0	1

- Furthermore, the Bluetooth polling algorithm requires that polling be performed using full length packets (i.e., at least one time slot T), which allows the nodes to acknowledge proper packet reception (or lack thereof) without additional overhead.
- Finally, it might be argued that this packet loss will cause the algorithms to mistakenly increase the number of active nodes. However, from the standpoint of the activity management algorithm, packet loss due to noise and interference does not differ — in qualitative terms — from the loss caused by congestion, that is, by buffer blocking at the intermediate nodes. As long as the thresholds and parameters of the algorithm are properly adjusted, the algorithms are able to maintain the received reliability within the desired limits.

2.8 Performance: Relative and Absolute Reliability

Simulations were carried out to explore the behavior of the relative reliability observed at the sink for each source piconet when the sleep management scheme is applied in the entire scatternet. Figure 2.9 presents the relative reliability observed at the sink for packets sent from slaves in P_4 (which is an exterior piconet) for packet arrival rates of 0.002, 0.003, and 0.004. All other parameters were set to the same values as before. Because all piconets operate under the sleep management scheme (except P_4, which is the subject of experiment), the relative reliability at the sink is

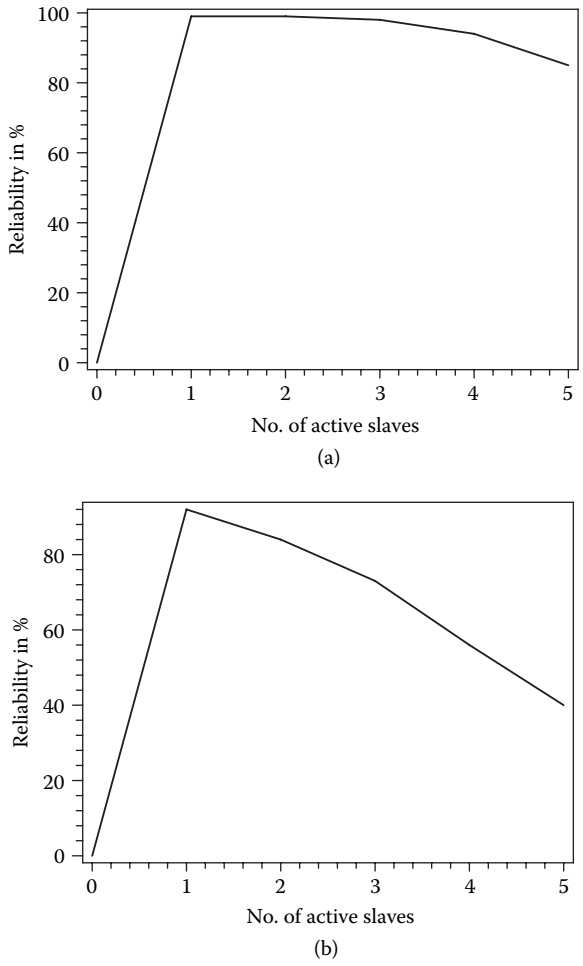


Figure 2.9 Relative reliability at the sink for packets from the slaves in piconet P_4 versus the number of active slaves in P_4: packet burst arrival rates of (a) 0.002 packets per slot, and (b) 0.005 packets per slot.

much higher than in the case when only one piconet uses the scheme; the peak value exceeds 95 percent under a wide range of packet arrival rates. Note that the event reliability for P_4 remains within limits T_L and T_H for one to two active slaves. The average number of active slaves for which the event reliability is within the limits decreases with the packet burst arrival rate, which is expected.

Figure 2.10 presents the analogous dependency, only this time the relative reliability corresponds to packets sent to the sink from slaves in P_2,

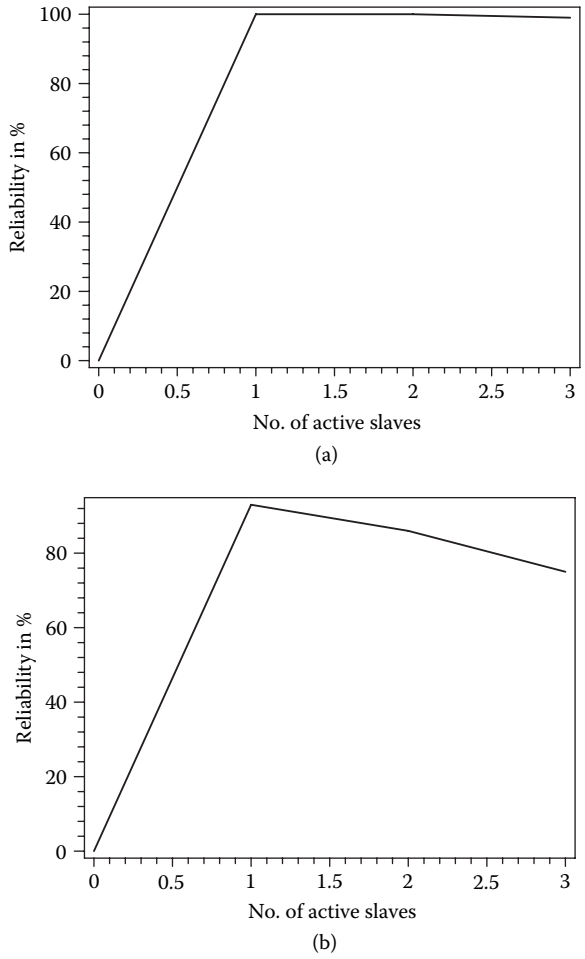


Figure 2.10 Relative reliability at the sink, for packets from the slaves in piconet P_2 versus the number of active slaves in P_2: packet burst arrival rate (a) 0.002 packets per slot and (b) 0.005 packets per slot.

which is an interior piconet. Because the same congestion control mechanism is used in all the piconets, the shape of the dependencies is almost identical to those from the previous set of diagrams.

To calculate the dispersion of the relative reliability, its mean, variance, and standard deviation are calculated for the data in Figures 2.9 and 2.10. We note that the increase in arrival rate leads to a decrease in mean value and an increase in variance, which can be used to indicate serious congestion. Figure 2.11 shows the development of the number of active slaves in P_4 over time, including the warm-up period of the simulator. The packet

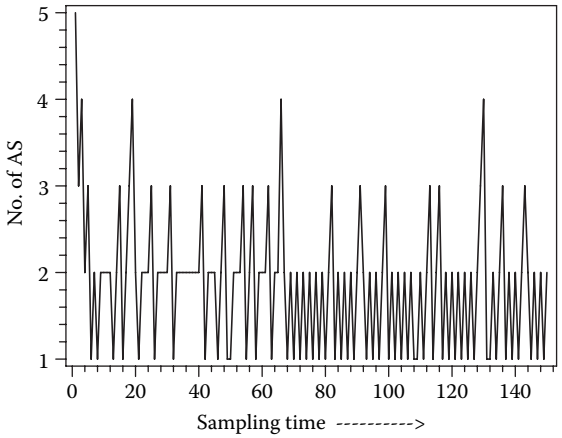


Figure 2.11 Fluctuation in the number of active slaves (AS) for P_4.

burst arrival rate was set to $\lambda = 0.002$ packet bursts per Bluetooth time slot. We note that the algorithm maintains average number of active slaves around 1.9.

Figure 2.12 and Figure 2.13 show absolute reliability observed at the sink for packets originating from slaves in piconets P_4 and P_2, respectively. At lower packet burst arrival rates, the absolute reliability is a monotonically increasing function of the number of slaves. While this result differs from the corresponding dependencies of relative reliability from Figures 2.9 and 2.10, keep in mind that the sleep management scheme was designed with the goal of maintaining the *relative* reliability, not its absolute counterpart, within certain limits. Of course, congestion control could be designed the other way around, that is, by specifying the desired absolute reliability and trying to achieve it with the highest possible relative reliability, as shown in Section 2.6.

2.9 Performance: Packet Loss at the Bridge Buffers

Another sign of congestion (and, by extension, a decrease in reliability) is the increase in packet loss rates at the bridge buffers. We have measured packet loss rates in our scatternet using the same setup as above: the sink was in piconet P_1, the desired reliability was set to 60 percent, the packet burst arrival rate was set to 0.005 (bursts per Bluetooth time slot), the bridge residence time was set to one piconet cycle, and polling parameters for slaves and bridges were $M_s = 3$ and $M_b = 12$, respectively. However,

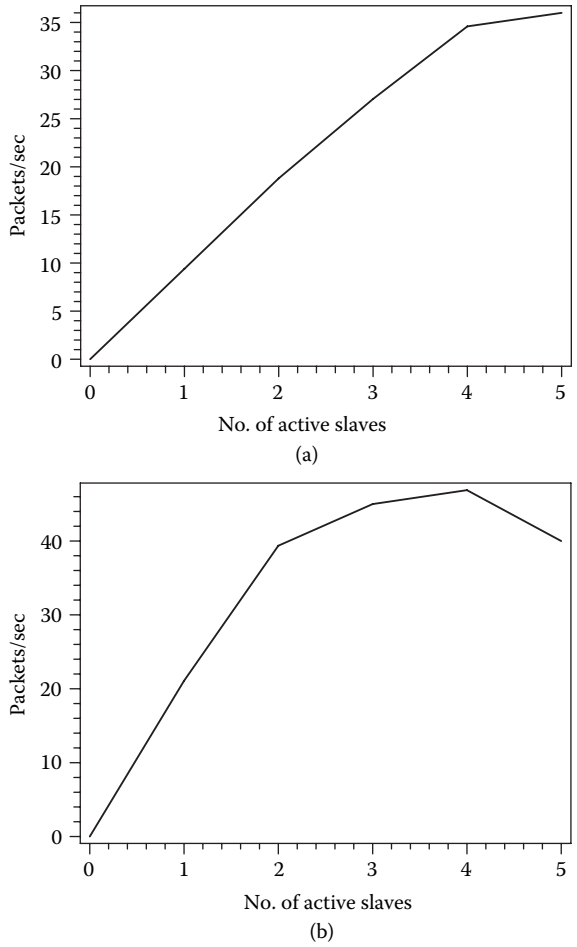


Figure 2.12 Absolute reliability at the sink for packets from slaves in P_4 versus the number of active slaves in P_4: packet burst arrival rate of (a) 0.002 packets per slot and (b) 0.005 packets per slot.

to get better insight, we varied the traffic locality probability in the range $P_l = 0.3 \ldots 0.8$ and the bridge buffer size in the range $8 \ldots 20$.

Packet losses at the buffers of bridges B_4, B_9, B_1, and B_3 are shown in Figure 2.14. Because of the symmetry of the network, B_4 and B_9 exhibit similar packet loss rates; we note that packet losses become significant for high inter-piconet traffic (i.e., with $P_l = 0.3$ and lower), which is characteristic of sensor networks. Similar conclusions hold for packet loss rates at the buffers of "interior" bridges B_1 and B_3 (which carry the data from

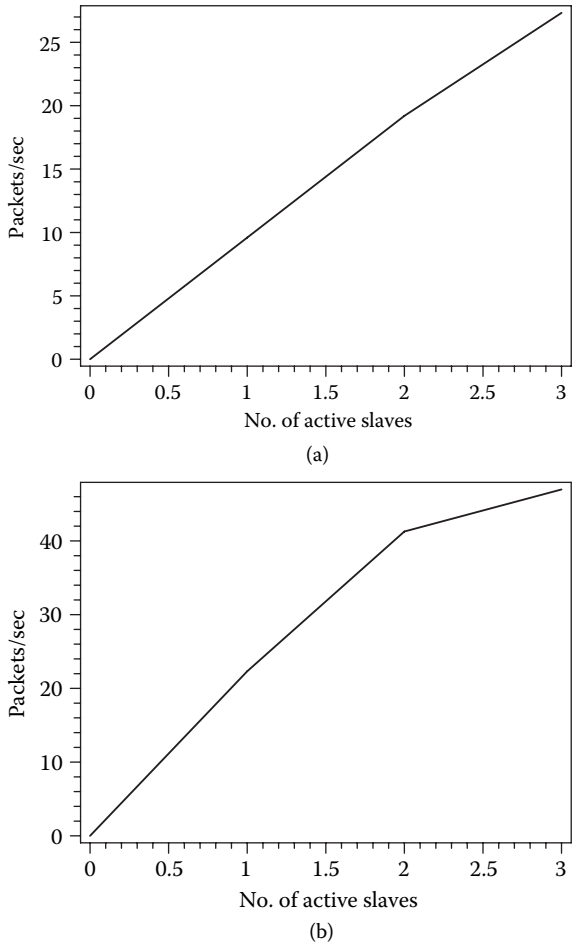


Figure 2.13 Absolute reliability at the sink for packets from slaves in P_2 versus the number of active slaves in P_2: packet burst arrival rate of (a) 0.002 packets per slot and (b) 0.005 packets per slot.

P_4, P_6, P_2, and P_3 to P_1), shown in Figures 2.14c and 2.14d. Because B_1 and B_3 carry data packets from two piconets each, the loss of data packets at these bridges is greater when compared to B_4 and B_9. We observe that under a realistic locality probability of $P_l = 0.3$, buffer sizes of 12 packets or more suffice to keep the packet loss very low; this offers a substantial advantage over the value of 40 or more, which is necessary in the network without sleep management [21].

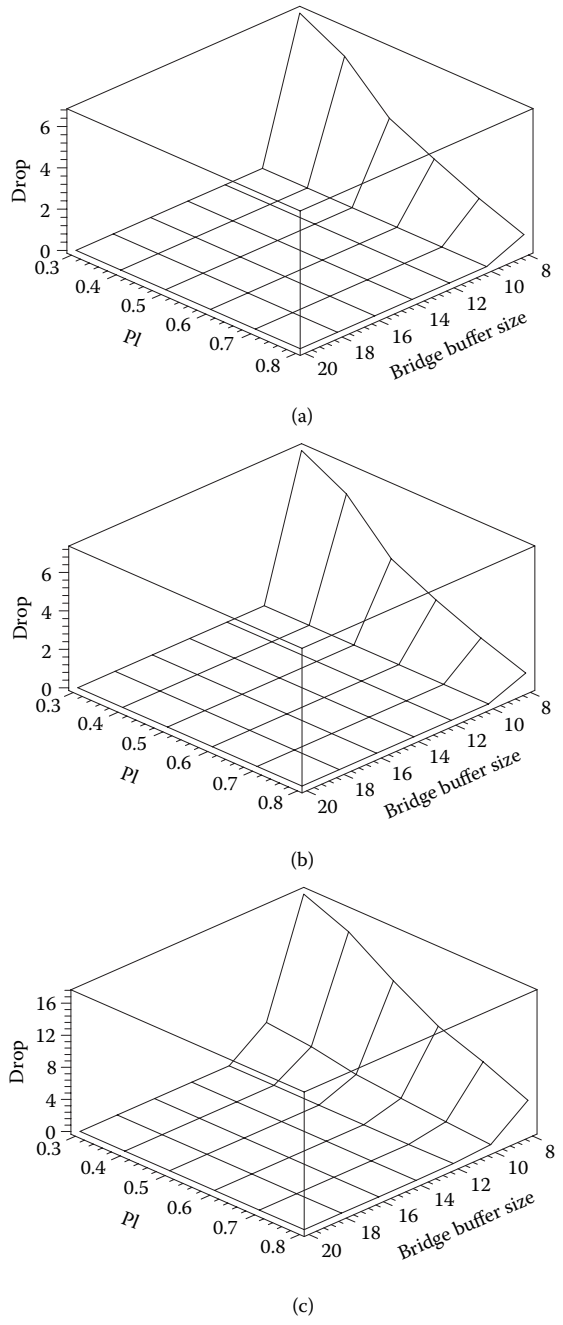
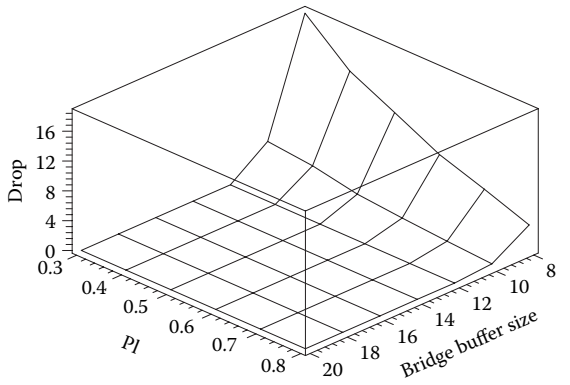
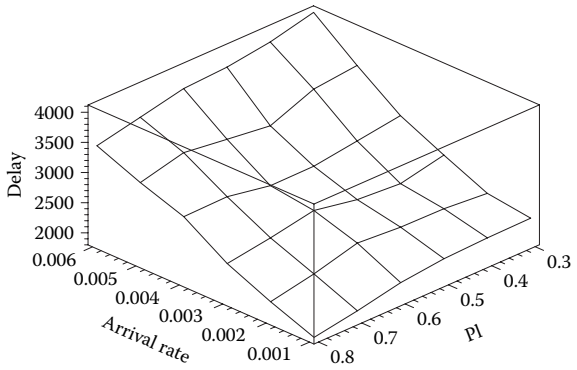


Figure 2.14 Bridge buffer drop rate (in percent) for various bridges at packet burst arrival rate of 0.005 packets per slot: (a) bridge B_4, (b) bridge B_9, (c) bridge B_1, and (d) bridge B_3.

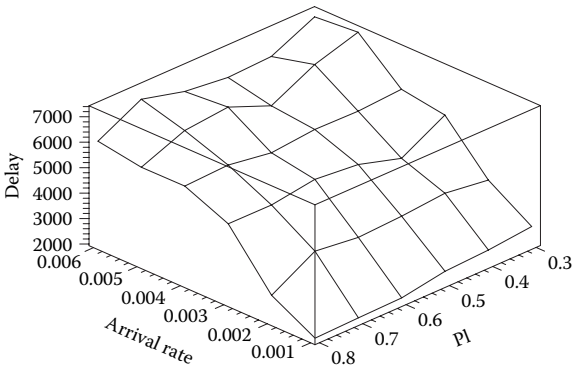


(d)

Figure 2.14 (Continued).



(a)



(b)

Figure 2.15 End-to-end packet delays: end-to-end delays for (a) traffic from P_6 to P_1 and (b) traffic from P_3 to P_1.

2.10 Performance: End-to-End Delay

Finally, end-to-end packet delays for traffic from P_6 to P_1 and from P_3 to P_1, are shown in Figures 2.15a and 2.15b, respectively. Both queueing and transmission delays are taken into consideration to calculate end-to-end delays. In this case, P_l varied in the range $0.3 \ldots 0.8$, and packet burst arrival rates were in the range $\lambda = 0.001 \ldots 0.006$. Because the interior piconets have more bridges than the exterior ones, their carried load is higher and so are the delays.

A note on simulations. All simulation results presented were obtained with a custom-built Bluetooth simulator implemented using the Artifex object-oriented Petri net engine by RSoft Design, Inc. [23].

2.11 Conclusion

In this chapter we have developed and evaluated two congestion control algorithms for Bluetooth-based sensor networks. Both algorithms are based on sleep scheduling of Bluetooth slaves. The first algorithm maintains required (fixed) event reliability at the sink using the minimum slave activity. It uses precalculated activity values obtained from the analytical and simulation models of the network.

The second algorithm keeps the whole network within the acceptable range of packet losses while maintaining minimum slave activity. In this case, source piconets use the information measured at the sink to regulate the activity of their slaves. Simulation results confirm that this sleep management policy results in decreased bridge buffer loss rates in all downstream bridges toward the sink, which means that bridges can be designed with smaller buffer space.

References

[1] M. Achir and L. Ouvry. Power Consumption Prediction in Wireless Sensor Networks. In *ITC Specialist Seminar on Performance Evaluation of Wireless and Mobile Systems*, Antwerp, Belgium, August 2004.

[2] O.B. Akan and I.F. Akyildiz. ESRT: Event-to-Sink Reliable Transport in Wireless Sensor Networks. In *IEEE/ACM Transactions on Networking, 13(5), 1003–1016*, October 2005.

[3] I.F. Akyildiz, W. Su, Y. Sankarasubramaniam, and E. Cayirci. Wireless Sensor Networks: A Survey. *Computer Networks, (Elsevier) Journal*, 38:393–422, March 2002.

[4] I.F. Akyildiz, M.C. Vuran, and O.B. Akan. On Exploiting Spatial and Temporal Correlation in Wireless Sensor Networks. In *Proc. WiOpt'04: Modeling*

and Optimization in Mobile, Ad Hoc and Wireless Networks, pp. 71–80, March 2004.

[5] J. Beutel, O. Kasten, and M. Ringwald. BTnodes — A Distributed Platform for Sensor Nodes. In *Proc. 1st Intl. Conf. on Embedded Networked Sensor Systems (SenSys)*, pp. 292–293, November 2003.

[6] Bluetooth SIG. *Draft Specification of the Bluetooth System*. Version 2.0, November 2004.

[7] A. Capone, R. Kapoor, and M. Gerla. Efficient polling schemes for Bluetooth picocells. In *Proc. IEEE Int. Conf. on Communications ICC 2001*, Vol. 7, pp. 1990–1994, Helsinki, Finland, June 2001.

[8] N. Golmie, R.E. Van Dyck, and A. Soltanian. Interference of Bluetooth and IEEE 802.11: Simulation Modeling and Performance Evaluation. In *Proceedings 4th ACM International Workshop on Modeling, Analysis and Simulation of Wireless and Mobile Systems*, pp. 11–18, Rome, Italy, July 2001.

[9] B. Hong and V.K. Prasanna. Optimizing System Life Time for Data Gathering in Networked Sensor Systems. In *Algorithms for Wireless and Ad-hoc Networks (A-SWAN) (Held in conjunction with MobiQuitous)*, August 2004.

[10] B. Hong and V.K. Prasanna. Optimizing a Class of In-Network Processing Applications in Networked Sensor Systems. In *1st IEEE International Conference on Mobile Ad-hoc and Sensor Systems (MASS 2004)*, October 2004.

[11] Standard for Part 15.4: Wireless MAC and PHY Specifications for Low Rate WPAN. IEEE Std 802.15.4, IEEE, New York, NY, October 2003.

[12] C. Intanagonwiwat, R. Govindan, D. Estrin, J. Heidemann, and F. Silva. Directed Diffusion for Wireless Sensor Networking. *ACM/IEEE Transactions on Networking*, 11(1):2–16, February 2003.

[13] N. Johansson, U. Körner, and P. Johansson. Performance Evaluation of Scheduling Algorithms for Bluetooth. In *Proceedings of BC'99 IFIP TC 6 Fifth International Conference on Broadband Communications*, pp. 139–150, Hong Kong, November 1999.

[14] P. Johansson, R. Kapoor, M. Kazantzidis, and M. Gerla. Rendezvous Scheduling in Bluetooth Scatternets. In *Proc. IEEE Int. Conf. on Communications ICC 2002*, pp. 318–324, New York, April 2002.

[15] Y.-Z. Lee, R. Kapoor, and M. Gerla. An Efficient and Fair Polling Scheme for Bluetooth. In *Proceedings MILCOM 2002*, Vol. 2, pp. 1062–1068, 2002.

[16] M. Leopold, M. Dydensborg, and P. Bonnet. Bluetooth and Sensor Networks: A Reality Check. In *1st ACM Conference on Sensor Networks*, November 2003.

[17] H. Levy, M. Sidi, and O.J. Boxma. Dominance Relations in Polling Systems. *Queueing Systems Theory and Applications*, 6(2):155–171, 1990.

[18] Z. Liu, Ph. Nain, and D. Towsley. On Optimal Polling Policies. *Queueing Systems Theory and Applications*, 11(1–2):59–83, 1992.

[19] J. Mišić and V.B. Mišić. *Performance Modeling and Analysis of Bluetooth Networks: Polling, Scheduling, and Traffic Control*. CRC Press, Boca Raton, FL, 2005.

[20] J. Mišić, K.L. Chan, and V.B. Mišić. Performance of Bluetooth Piconets under E-Limited Scheduling. Tech. Report TR 03/03, Department of

Computer Science, University of Manitoba, Winnipeg, Manitoba, Canada, May 2003.

[21] J. Mišić, V.B. Mišić, and G.R. Reddy. On the Performance of Bluetooth Scatternets with Finite Buffers. In *Proc. WWAN2005 International Workshop on Wireless Ad Hoc Networking (ISCDS'05 Workshops)*, pp. 865–870, Columbus, OH, June 2005.

[22] V.B. Mišić, J. Mišić, and K.L. Chan. Walk-in Scheduling in Bluetooth Scatternets. *Cluster Computing*, 8(2/3):197–210, 2005.

[23] RSoft Design, Inc. *Artifex v.4.4.2*. San Jose, CA, 2003.

[24] Hideaki Takagi. *Queueing Analysis*, Vol. 1: Vacation and Priority Systems. North-Holland, Amsterdam, The Netherlands, 1991.

[25] C. Wan, S.B. Eisenman, and A.T. Campbell. CODA: Congestion Detection and Avoidance in Sensor Networks. In *Proceedings of the First International Conference on Embedded Networked Sensor Systems*, pp. 266–279. ACM Press, November 2003.

[26] C.Y. Wan, A.T. Campbell, and L. Krishnamurthy. PSFQ: A Reliable Transport Protocol for Wireless Sensor Networks. In *First ACM International Workshop on Wireless Sensor Networks and Applications (WSNA 2002)*, pp. 1–11, September 2002.

[27] S. Zürbes. Considerations on Link and System Throughput of Bluetooth Networks. In *Proceedings of the 11th IEEE International Symposium on Personal, Indoor and Mobile Radio Communications PIMRC 2000*, Vol. 2, pp. 1315–1319, London, U.K., September 2000.