Sharing a Medium Between Concurrent Protocols Without Overhead Using the Capture Effect

Michael König
Distributed Computing Group
ETH Zürich, Switzerland
mikoenig@ethz.ch

Roger Wattenhofer
Distributed Computing Group
ETH Zürich, Switzerland
wattenhofer@ethz.ch

Abstract

Most wireless systems suffer from the lack of a dedicated control plane. High-priority control messages have to contend for medium access the same as any other message. Existing methods providing QoS guarantees in this setting rely upon opportunistic sending or on scheduling mechanisms, and as a result incur undesirable tradeoffs in either latency or impact on regular non-priority traffic.

We present a technique to simultaneously execute multiple protocols of different priorities, without compromising bandwidth or latency of regular traffic not affected by priority traffic. Using power control and moderately tight synchronization we exploit the capture effect to give each protocol almost complete access to the network's resources as long as no protocol of higher priority wishes to use them. We examine which impact the properties of the network graph and the capabilities of the wireless hardware have on the effectiveness of our technique.

We suggest an example scenario of a wireless sensor network of fire detectors, with a low-priority protocol collecting statistical data and confirming aliveness, while a high-priority protocol wishes to report fire alarms to a base station as quickly as possible. For this example application we deploy an implementation based on Contiki on a wireless testbed of TelosB motes and achieve near optimal latency and bandwidth for priority traffic while not disturbing low-priority traffic where it is separated sufficiently in space or time.

Categories and Subject Descriptors

C.2.1 [**Computer-Communication Networks**]: Network Architecture and Design—*Wireless communication, Distributed networks, Network communications*; C.2.5 [**Computer-Communication Networks**]: Local and Wide-Area Networks—*Access schemes*

General Terms

Algorithms, Design, Experimentation, Performance, Reliability

Keywords

capture effect, power control, medium access control, traffic prioritization, wireless sensor networks

International Conference on Embedded Wireless Systems and Networks (EWSN) 2016
15–17 February, Graz, Austria

ISBN: 978-0-9949886-0-7

1 Introduction

While the deployments of wireless networks continue to grow in number and size year by year, protocol designers are still struggling to understand how to most efficiently use the wireless medium. Not only is operational disruption through environmental noise caused by other networks or microwave ovens oftentimes unpredictable in urban settings, but due to the broadcast nature of wireless transmissions, nodes participating in a network frequently experience interference with nearby nodes of the same network.

To tackle this medium access problem given a fixed frequency spectrum, generally one of two approaches is used: (1) opportunistic sending (CSMA) or (2) scheduling (TDMA).

(1) Opportunistic sending works by a first-come-first-serve philosophy and hopes for a free channel at the time of traffic emergence, sending immediately, asking questions later. After sending, an explicit or implicit acknowledgment from the recipient is required to determine whether the transmission was successful or needs to be repeated. Variants of opportunistic sending such as clear channel assessment (CCA) and request-to-send/clear-to-send (RTS/CTS) have been proposed to reduce the number of haphazard collisions, but suffer from hidden or exposed terminal problems and introduce overhead in terms of packets sent and latency, which especially in the case of RTS/CTS may quickly grow very noticeable [26, 21, 15].

(2) The alternative is to employ one or more nodes with the role of scheduling authorities. These nodes manage the permissions to send packets in their immediate network neighborhood. Each node is either allotted a periodically recurring time slot for sending, or may need to first explicitly request a reservation for the channel from its local scheduler(s). This approach may completely avoid collisions during regular operation but incurs latency and possibly also bandwidth penalties. While generally opportunistic sending is preferred for its simplicity and flexibility in most scenarios, the scheduling approach has also found its way into

widely deployed systems such as Bluetooth [8].

Clearly neither approach is optimal in all scenarios, nor is every scenario served well by either approach. For example, consider the case of an emergency signal needing to travel to a destination node in as little time as possible. Using opportunistic sending, progress may stall almost indefinitely when the network is under heavy load, while a scheduling approach might reserve every second slot for emergency messages which guarantees arrival in twice the minimum possible time at the cost of halving the number of slots available for regular non-emergency traffic. Hybrid MAC layers such as Z-MAC [22] have been proposed, with the goal of combining the advantages of CSMA and TDMA. Z-MAC realizes this by dynamically switching between CSMA and TDMA based on network load.

Network mechanisms designed to ensure fairness or more specifically offering guarantees about network performance are grouped under the term quality of service (QoS). While QoS research for wireless networks is an area with a wealth of history, certain aspects of wireless networks are yet to be fully understood and utilized. Among these is the so-called *capture effect* (also known as *physical layer capture*). For a long time, protocol designers tried to avoid collisions whenever possible, working under the assumption that any collision of wireless packets at a receiving node inevitably leads to the failure of that node to decode any of the messages. This loss rate has been shown to have been significantly overestimated [25, 23]. This is due to the capture effect, a phenomenon which oftentimes allows the receiver of a wireless transmission to continue correctly decoding the transmission, in spite of interference caused by other transmissions starting during the original transmission.

We consider a "protocol" to be a self-contained distributed algorithm using the network to transmit messages, coping with lost messages and usually avoiding collisions where possible. In this paper we propose a technique to "layer" such protocols of different priority levels on top of each other using the capture effect, effectively enabling a priority process to use almost all the resources of a network, while at the same time allowing lower level processes separated from the priority traffic in space and/or time to use the network at no additional overhead. Note that giving unrestricted resource access to a protocol necessarily implies that it may starve all lower-priority protocols. We also propose a mechanism to only administer a share of the resources to a protocol, but this unavoidably introduces a latency overhead.

We require a certain degree of clock synchronization (clock difference below 160 µs between any pair of nodes with distance at most two hops) to be able to make best use of the capture effect, and impose the notion of time slots on the network. Hence, we assume that at least one of the layered protocols contains a component periodically resynchronizing all nodes. Furthermore, we require the wireless hardware to offer transmission power control, which is a common feature even among older hardware.

The basic idea is to cause the capture effect whenever a node would receive multiple packets in the same time slot. This is done by choosing the transmission power of each node such that it falls into one of multiple pre-computed bands of reception power at the intended receiving node. Given these bands are separated well enough, the receiving node will almost always (in over 98% of cases) be able to decode the packet in the strongest band without error. This implicit prioritization lets us avoid the overhead caused by more explicit measures such as schedules. On the other hand, there are some inherent disadvantages tied to our technique, namely the need for time slotting and the predetermination of transmission powers, removing the ability to intentionally save energy on short links and to save hops with long links requiring the highest possible transmission power.

We specifically target wireless sensor networks (WSNs), which typically form networks with a relatively large connectivity graph diameter and favor node quantity over advanced wireless capabilities. Our technique accommodates these conditions particularly well. For example, as higher layer protocols only disturb their immediate neighborhood in the connectivity graph, more spread out networks are more likely to benefit.

We tested our technique on an example alarm reporting protocol. Our implementation is based on Contiki running on a public testbed of Tmote Sky sensor nodes (also known as "TelosB") [20]. We achieve near optimal alarm reporting latency and almost no packet loss on the high-priority layer, while when under load indeed causing comparatively little disturbance to the underlying low-priority traffic which we use to ensure node liveness and keep the nodes' clocks synchronized.

2 Related Work

Already in 1976, the capture effect in FM receivers was modeled by Leentvaar et al. [11]. To combat it they proposed using bandlimiting at the receiver. The capture effect is not a phenomenon limited to FM transmissions. Ash [1] showed that it is possible to obtain an equivalent and even stronger effect in AM receivers.

While the capture effect had at first been considered undesirable, it was soon ascribed inadvertent performance boosts in common wireless scenarios such as slotted ALOHA [4] and everyday 802.11 traffic [15, 25]. Soon, a number of environmental influences like noise, path loss, shadowing and fading were identified to be contributing to the capture effect's potency as a general packet reception enhancer [2, 13].

Other research was conducted on the details of packet timing, as common transceiver hardware does not facilitate switching reception from one packet to another mid-demodulation. Thus, if a much stronger packet starts during the reception of a weaker one, both packets are lost (save for the leading portion of the weaker one). A well-studied quirk of the capture effect is that it may occur even when the stronger signal arrives after the weaker one, as long as it still arrives before the end of the preamble of the weaker signal [9, 25, 23, 27].

The obvious solution to the "stronger packet arrives too late" problem is to continuously scan the medium for transmission preambles, even during packet reception. This requires more specialized hardware support, but has nevertheless already been thoroughly investigated [9, 25, 17, 10]. To make best use of the capability to switch to stronger pack-

ets during reception (also known as "message in message"), Manweiler et al. [17] discuss how careful ordering of transmissions enables the parallel utilization of traditionally conflicting sender-receiver pairs.

When it comes to low-power wireless networks such as those comprised of sensor nodes, the meticulous study by Son et al. [23] provided a solid foundation. While they find that occurrence of the capture effect can be guaranteed given a large enough SINR value, they find a significant gray region of up to 6 dB to exist in practice. Further, they find the SINR threshold to be heavily dependent on the transmitting hardware and the selected transmission signal strength. Yuan et al. [27] continue this study and propose a packet reception model for concurrent transmissions, including the special case of constructive interference.

Nyandoro et al. [18] consider the scenario of a 802.11 access point and several clients split into low-priority and high-priority clients. They propose using a significantly higher sending power for the high-priority clients. Due to the capture effect collisions between packets from high and low-priority clients will then always be solved in favor of the high-priority client. Patras et al. [19] go as far as to suggest deliberately fluctuating sending power levels in order to make the capture effect more likely to occur in case a collision takes place. They show that in practice this can translate to throughput gains of up to 25%. As links become more heterogeneous, though, this effect decreases, and instead an increase in fairness can be observed.

Lu et al. [16] implement a flooding protocol called *Flash*, which ignores collisions between neighboring nodes at the flooding front, leaving it to the capture effect to let each node receive a copy of the message sooner or later. They show that such recklessness can in fact improve latency by as much as 80%.

Various approaches to coordinate simultaneously running protocols competing for medium access have been suggested. Flury et al. [5] proposed "slotted programming", dividing time into slots and assigning every protocol a fixed portion of the slots. This framework is implemented in a fashion transparent to the protocols, effectively making them independent and modular building blocks for larger systems. Note that in contrast to the method presented in our paper, slotted programming incurs a significant penalty on the total throughput when protocols are unable to make use of the scheduled slots assigned to them.

The same work also includes a proposal for an alarm mechanism: Flury et al. [5] suggest alarmed nodes transmit a specific waveform at maximum power. Other nodes, upon detecting the waveform, become alarmed and start transmitting the waveform as well, thus spreading the alarm. The authors find that, even without synchronization between the nodes, the collision of the signals is not detrimental to the spread of the alarm. However, extending this scheme to alarm signals carrying more information than the alarm's presence itself appears to be difficult. Additionally, these alarm signals are undirected and will prevent any regular traffic in the network.

Cidon et al. [3] propose establishing a control plane for Wi-Fi networks by inserting high-power "flashes" into regular packets. These flashes are a waveform of far higher amplitude than the rest of the signal and are added to regular data symbols, effectively erasing those symbols. By exploiting the underlying OFDM encoding, which sends multiple redundant copies of each data bit either separated in time or frequency, they are able to insert on the order of 50,000 flashes per second without causing a packet loss rate of more than 1%. The occurrence and spacing of these flashes may then be chosen to represent out-of-band data. While this approach does not require additional frequency bands or time slots for control messages, its main disadvantage is its reliance on specialized hardware.

For the specific scenario of time-critical alarm message propagation, Li et al. [12] propose incorporating slots allocated for emergency messages into a regular scheduling mechanism, but to employ slot stealing to avoid wasting network bandwidth in the absence of emergencies. A short while after the start of a slot assigned to emergency messages, if the slot is detected to remain unused, nodes may steal and use the remainder of the slot to send regular traffic. They further provide a simulation framework tailored to such wireless alarm systems. In constrast to our work, their method relies on an explicit scheduling mechanism to designate recurring slots for emergency messages. This incurs an overhead in latency and does not scale well to a larger number of priorities, as the time required to detect slot use grows and thus further erodes the concept of time slotting.

A different approach, employed to great effect by cellular networks technologies such as LTE, is to send and receive on multiple different frequency bands, allowing to use a subset of them as an independent control plane [6]. In contrast, we do not consider the use of multiple frequency bands and restrict ourselves to simple wireless transceivers able to send and receive on a single band at a time only.

3 The Capture Effect

In this section we will go into detail about the capture effect and its inner workings, as it is integral to the method we propose. Furthermore, we will discuss the exact parameters for its occurrence we measured on the hardware and testbed we will be using for our example implementation.

The *capture effect* is a term describing the general phenomenon of wireless receivers being able to decode the strongest of multiple signals without error, effectively completely ignoring the weaker signals. Standard wireless hardware is designed to send and receive wireless data strings in the form of discrete packets, which may be inserted into the noisy carrier medium at arbitrary points in time. Due to this, harnessing the capture effect on such hardware is limited to a certain set of scenarios.

Typically, wireless receivers use specific pre-defined chip patterns to detect the start of a transmission. These patterns are called a *preambles* or *syncwords* (short for synchronization words). They serve multiple purposes: For one, they allow to, with a high probability, identify a starting transmission amongst the environmental noise and hence avoid mistaking noise for a transmission even in settings where transmissions create signals barely stronger than the noise. Another purpose, whose side-effect is particularly important

Table 1. Listing of the possible outcomes of two packets arriving at the same receiver simultaneously. Δ specifies the time difference $t_A - t_B$ between the arrival times of the two packets and d_i denotes the duration of packet i. τ_1 and τ_2 denote two thresholds between which the probability of a correct reception of packet A tapers off.

	Packet A (strong)	Packet B (weak)
$\Delta < -d_A$	correct	correct
$-d_A < \Delta < 0$	correct	not at all
$0 < \Delta < \tau_1$	correct	not at all
$\tau_1 < \Delta < \tau_2$	tapering chance	not at all
$\tau_2 < \Delta < d_B$	not at all	partially corrupted
$d_B < \Delta$	correct	correct

for summoning the capture effect, is the aligning of the receiving wireless transceiver's internal clock with the phase of the signal. This essentially means that once a receiver has detected a preamble, it can configure itself to easily decode the following data symbols and stream their values into some kind of memory.

As a result, upon hearing a preamble, receivers effectively commit to receiving a particular transmission, locking their clocks to that transmission's phase and often also its length (which in many physical layer protocols is transmitted amongst the first few data symbols). This behavior is especially desirable when bursts of noise frequently occur in the environment, but partially corrupted packets may still be valuable, either due to error correcting codes or simply full data integrity not being a critical requirement.

When two or more signals can be heard at a receiver simultaneously, they act as noise to each other, i.e., cause interference for one another. Generally, it is impossible to decode the weaker signal(s), unless the hardware is capable of more advanced techniques, such as decoding and subtracting the stronger signals first or using a coding scheme such as CDMA. However, this paper is aimed at scenarios employing simple sensor nodes and does not rely on such functionality. Hence, we will assume that when a stronger transmission starts during the reception of a weaker transmission, the weaker transmission is certain to become corrupted.

On the other hand, when the stronger transmission starts first, it can nearly always be received completely without error. This is in spite of the fact that a prediction based purely on the signal powers and the classical SINR model will conclude that data corruption would occur for a significant range of power differences. The locking onto the phase of the signal effectively diminishes the influence of competing transmissions and lowers the SINR threshold required to be met for correct reception.

Due to the nature of the use of preambles, the requirement, that the stronger transmission comes first, is significantly eroded: the weaker transmission may come first, as long as its preamble is not completely received before the preamble of the stronger transmission begins. This happens because the stronger preamble destroys the end of the weaker preamble, hence, the receiver no longer considers the weaker signal to be a valid packet. Further, in some scenarios the

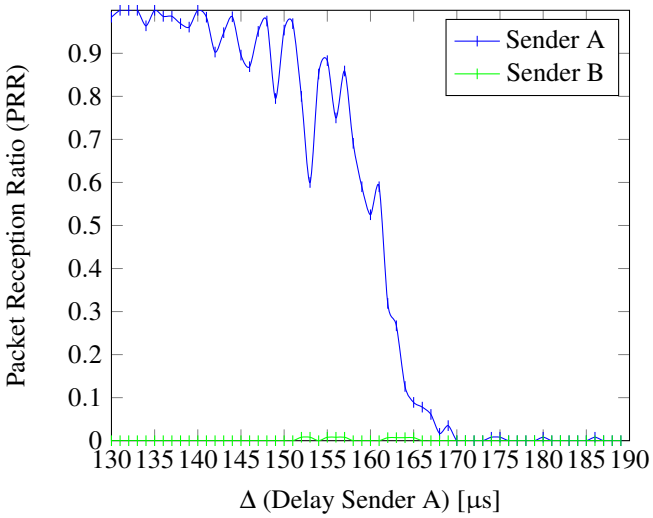

Figure 1. Occurrence probability of the capture effect at a receiver plotted against the time difference between the two incoming packets. Sender A's signal was significantly stronger averaging at -69 dBm RSS (Received Signal Strength), compared to Sender B at -82 dBm RSS. 130 samples were taken per delay value.

window for the stronger transmission to arrive has been reported to extend even into the first 3 bytes of the weaker transmission [27].

A summary of the outcomes in each possible scenario for 2 packets can be seen in Table 1. Of special interest here are the constants τ_1 and τ_2, which dictate the timing thresholds required for the capture effect to occur. We expect these to closely match the length of the preamble.

To find the exact values of τ_1 and τ_2 for our specific hardware and testbed setup, we conduct a few experiments pitting two senders against each other to try to successfully transmit a packet to a single receiver. To mimic actual protocol performance as close as possible we only use the hardware's innate ability to keep time and synchronize clocks. Hence, the receiver periodically sends a packet both senders use to synchronize their local clocks to the receiver's. The period is chosen to keep the clock error strictly below $\pm 0.5\,\mu s$, which is close to optimal considering the clocks' frequency of 4 MHz. The details of achieving such synchronization precision are beyond the scope of this paper. In the remaining slots both senders send packets with varying transmit power levels or delays.

Figure 1 shows the results for one sender using a significantly larger sending power and, due to links of similar quality being used, significantly higher received signal strength. We observe the probability of the receiver capturing the stronger packet to fall to zero roughly over the interval from $\tau_1 \approx 150\,\mu s$ to $\tau_2 \approx 165\,\mu s$. At the bitrate of 250 kbit/s 160 µs corresponds to 10 symbols or 5 bytes, which matches the length of the preamble plus the immediately following SFD (start of frame delimiter) byte. The spread $\tau_2 - \tau_1 \approx 16\,\mu s$ matches the length of one symbol and

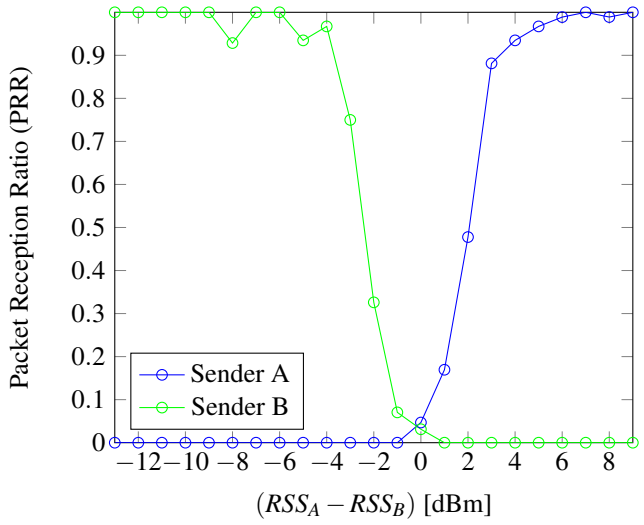

Figure 2. Occurrence probability of the capture effect at a receiver plotted against the power difference between the two incoming packets. Sender A's transmit power was varied over its whole available range, producing RSS_A values from −89 dBm to −68 dBm, while Sender B's transmit power was kept constant, producing RSS_B values from −79 dBm to −76 dBm. 40 samples were taken for each of the available 32 transmit power values, resulting in a total of 1108 comparable instances, i.e., instances in which we were able to measure both senders' RSS values.

is roughly centered around $\Delta = 160\,\mu s$.

It is worth noting that the study by Yuan et al. [27], also using TelosB motes, measured the transition window to be $\tau_1 \approx 128\,\mu s$ to $\tau_2 \approx 224\,\mu s$, resulting in a spread of roughly 6 symbols or 3 bytes. We are not completely certain about the cause of this discrepancy, but presume the difference in path quality to play a significant role: we used 3 nodes part of the testbed, located in different rooms, while they had placed the nodes in clear line of sight at a distance of 1 meter.

We also measured the effect of signal strength difference on capture probability. Figure 2 shows our results for the case of both senders sending simultaneously ($|\Delta| < 1\,\mu s$), but one sender varying its transmission power. To determine the difference in RSS and counteract the fluctuation of the link qualities over time, we additionally measured each sender's RSS value within at most 0.5 µs of each sample. We observe the gap $thres_{power}$ between either of the senders having their packets captured with a high probability, say $PRR > 90\%$, to be about 5 to 7 dBm. This roughly matches the values cited in existing literature, e.g., [23]. It is worth noting, however, that the gap size $thres_{power}$ likely depends on the environment, since in most existing models a higher noise floor implies needing a higher signal power to beat the SINR threshold required for successful capture. Further, we find that the probability of either packet to be received successfully drops to around 5% when the signal strengths are identical.

In conclusion, we observe that in order to be able to reliably call on the capture effect in a slotted setup (see next section), competing senders should have a clock difference below 160 µs and a power difference above 5 dBm. This level of synchronization we can achieve by synchronizing about once every few minutes.

4 Layering Protocols

4.1 Slot Logic

Traditionally, executing multiple protocols in parallel is likely to incur a penalty on the utilization of the network resources and/or the performance of the protocols themselves. For example, if two protocols access the medium alternatingly using time slots, up to half the slots may be wasted, while one of the protocols is idle and the other has demand for more than its share of time slots. Further, in this scenario, information propagation latency is doubled, as no information can leave a node sooner than 2 slots after its arrival. Using more opportunistic approaches, such as when using CSMA/CA (Carrier Sensing Multiple Access with Collision Avoidance), the problems mentioned above do not occur: any number of idle protocols do not influence the network's utilization or the performance of the other protocols. However, when the load on the network becomes too large, unfairness and starvation become threats to effective operation.

In this section we will detail our proposed method of parallelizing, or *layering*, k protocols with the aim of combining the benefits of the approaches mentioned above: no unused network resources under load, while also offering fairness and prevention of starvation. Further, our method allows prioritization of protocols, giving higher-priority protocols almost complete access to the network's resources at the cost of possible starvation of lower-priority protocols in areas of the network not sufficiently separated from the high-priority traffic in space or time. Finally, this section will discuss how network topology and environment influence the number of layers our method can support.

The core idea is to deliberately provoke the capture effect at every node whenever it is destined to receive multiple packets at the same time. To do so, we enforce time slotting and require the clock difference between any two nodes within a nodes immediate neighborhood to be below τ_1 (see Section 3). By ensuring all potentially competing packets start within a time interval of length τ_1, we obtain that which packet is to be received in a time slot is solely dependent on the arriving packets' signal strengths, but not on their relative timings. Given a sufficient spread in signal strengths, the capture effect is almost certain to enable successful reception of the packet with the strongest signal. We will discuss why this assumption is a reasonable one to make below.

The next piece of the scheme is to use transmission power control at each sender to specify the "layer" of each packet. As nodes may have varying distances and link qualities to each other, the transmission power cannot simply be derived from the layer of the packet to be sent, but must consider the destination node. In effect, for every receiving node a set of incoming signal strength intervals needs to be chosen, different enough to be distinguishable by the capture effect, but similar enough to fit within the range of signal strengths each of the neighboring nodes can produce. Thus, for every sending node, for each of its neighbors and for each of the

layers the correct sending power needs to be chosen.

Algorithm 1 Pseudocode for Slot Logic

```
out ← ∅
for all protocols P_l with layer l ← 1 to k do
  P_l.compute_slot()
  if P_l.outgoing_packet ≠ ∅ then
    out ← P_l.outgoing_packet
    P_l.outgoing_packet ← ∅
  end if
end for
if out ≠ ∅ then
  Transmit out this slot (using the correct power for out's
  target and layer)
else
  Listen this slot
  in ← incoming_packet
  if in ≠ ∅ then
    P_in.layer.process_packet(in)
  end if
end if
```

Finally, every protocol is uniquely assigned to a layer. We label the layers $1,\ldots,k$, where the protocol of layer k has the highest priority and the protocol of layer 1 has the lowest. Every slot, every node executes Algorithm 1: First, it performs each protocol's slot computation separately, while storing the packet the highest layer protocol wants to send (*out*) and discarding all others. If any packet was chosen this way, it is sent at the sending power corresponding to its destination and protocol layer. If no packet was chosen, the node listens for the duration of that slot and delivers any received packet to the correct protocol.

This setup attempts to give each protocol the illusion of being the only protocol present. This is achieved by protocols experiencing a "packet loss" if a protocol of higher layer is active at the same time: if a higher layer is overriding the sending of a lower layer packet, that lower layer packet simply appears to have been lost in transit; conversely, if a packet of a higher layer is overriding the reception of a lower layer packet, that packet's fate appears indistinguishable from true packet loss as well. As occasional packet loss is a common occurrence in almost every environment due to noise bursts or interference, most wireless algorithms are innately capable of recovering from a loss of packets. Hence, they are perfectly suitable to be used as lower layer protocols. The highest layer protocol experiences no packet loss due to the presence of other layers (with one exception noted below), but is still subject to the usual environmental impediments. If the environment is in fact controlled enough to not suffer any such packet loss, as might be the case in clinical settings such as perhaps data centers, a protocol relying on a low packet loss ratio may be used as the highest layer.

We do not consider queuing packets from multiple protocols desiring to send from the same node in the same slot, as this would tamper with possible protocol-internal slot schedules of protocols whose packets have been delayed. This would damage the illusion, and require significant changes to the way protocols for use in the lower layers are designed, such that common known protocols can no longer easily be used. Simulating packet loss is hence a cleaner solution, while the option of allowing layering-aware protocols to immediately know if their packet was dropped, such that they may queue it for the next slot if desired, is still available.

There is one scenario, however, in which the illusion inevitably breaks down. As we are assuming that the wireless hardware is not capable of receiving while transmitting, a problem occurs when a node is choosing to send in a slot due to a protocol of layer i, but would in the same slot receive a packet on layer $j > i$. Here the protocol of layer j will experience packet loss due to a lower layer protocol. Unfortunately, it is impossible to prevent this scenario from occurring without also introducing significant overhead to all other scenarios: if the traffic demand on layer j can occur spontaneously, for instance, to propagate an alarm event, every node's layer i protocol may be in any state, including having chosen to send in that particular slot. If one forces layer i a priori to not send in certain slots, the latency and bandwidth penalties tied to TDMA are inevitable.

We found that for applications, in which the highest level protocol aims to achieve the lowest latency possible (such as the example application discussed in Section 5), a reasonable workaround is to send every high-priority packet twice in successive time slots. Note that while this does double the amount of packets sent, the latency only increases when the described scenario indeed occurs. Running common algorithms, a node that is sending in slot t is unlikely to send again in slot $t+1$ as there would not have been any input in slot t to instigate another outgoing transmission. This is even true if multiple layers wanted to send in slot t, as all their packets would have been either sent or discarded in slot t. Hence, a high-priority packet sent in two successive time slots is bound to arrive in at least one of the two slots.

4.2 Power Choices

The main difficulty now lies in ensuring a good spread in the signal strengths of all packets received at each node in the same slot. One assumption we make is that the individual protocols avoid causing multiple of its packet to collide at the same node. This is reasonable especially for protocols following the traditional school of thought, which dictates all simultaneous packet arrivals to be fatal collisions. Given this assumption and the fact that every protocol runs on its own unique layer, all packets arriving at a node in the same slot belong to different layers and should thus have sufficiently different signal strengths for the capture effect to be able to enable reception of the strongest packet.

There exists a tradeoff between the number of available layers (and thus number of parallelizable protocols) and the achievable spread of received signal strengths at each node. The network topology and in particular the homogeneity of the network's link qualities play a large role in enabling a higher number of layers to be well-separated at each node. "Well-separatedness" requires the difference in received signal strength between every pair of layers to exceed the threshold of $thres_{power}$ (which we found to be at least 5 dBm on our testbed, see Section 3). We found that in a perfectly homogeneous setting where every link is either

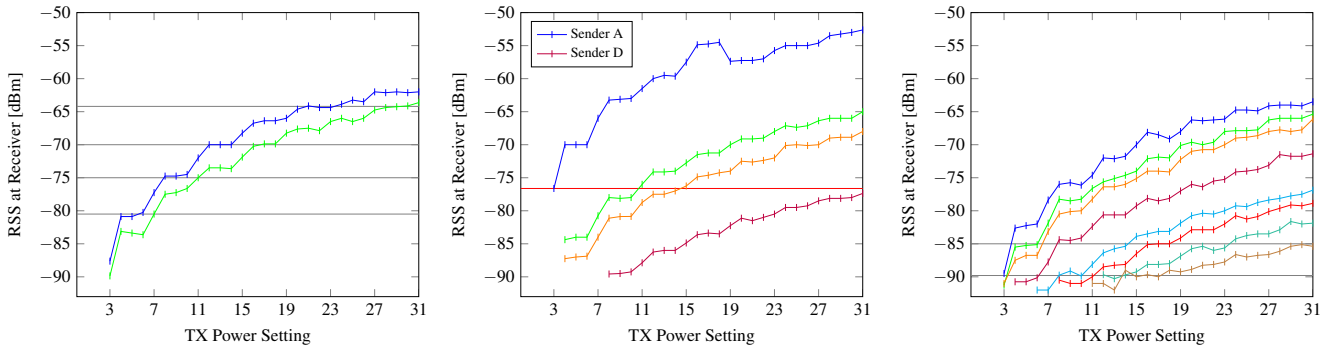

Figure 3. Shown are the RSS (received signal strength) values for different output powers for different links at the same receiver.
(a) Two links of similar quality easily allow 4 layers to be differentiated (horizontal lines). A possible fifth layer could identify with powers below -88 dBm.
(b) An example of a node with an extremely short distance neighbor (Sender A). No power setting allows a packet from Sender D to be captured while Sender A is sending as well.
(c) An example of a typical node with no very close neighbors. 2 layers are supported by all links.

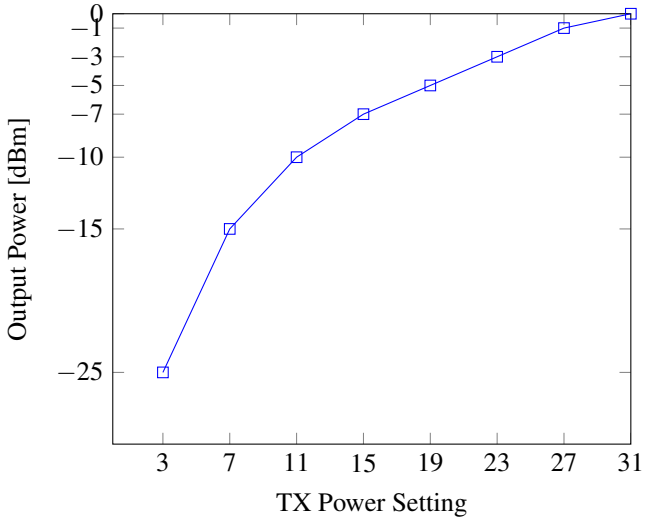

Figure 4. The 32 available output power settings on the CC2420 wireless transceiver [24]. Note that half the available values cover only a 7 dBm interval and settings below 3 are not usable.

of high quality (high range of powers usable for successful transmissions) or not of significant power, the number of available layers becomes maximal. Using our hardware we found up to 4 or 5 sending powers to be distinguishable at a receiver, see Figure 3(a). Such a high number of layers (4 or more) is likely only feasible in settings with a high degree of control over node positioning and environmental influences.

More commonly, networks contain varying levels of heterogeneity, with some areas containing only long-distance/low-quality links, some areas more tightly packed with low-distance/high-quality links, and many areas being cases in between. For our hardware, especially these in-between cases spell trouble due to the granularity of selectable sending powers decreasing sharply as power values decrease, see Figure 4. As a direct result a bottleneck for the number of layers forms at nodes with both long-distance and short-distance links. In the example of Figure 3(b), the low-quality link of Sender D can only provide receive powers in the range from -89 to -78 dBm, a range which Sender A cannot reach even with one power setting. In Figure 3(c), while no extremely high-quality links are included, the higher-quality links still offer only a very low power granularity in the range feasible for lower-quality links.

We find that for our hardware 2 clearly distinguishable layers are possible in essentially all topologies, but identify the occurrence of both “long” and “short” links at a single node as the main bottleneck. Note that the cause for the bottleneck is not present in nodes which have only long or only short links, as in these cases the links’ power ranges overlap very well. Essentially, the smaller the upper bound on the difference between the longest and shortest link at any node in the network, the higher the number of available layers.

When faced with the problem of the network supporting too few well-separated layers, there are several possible solutions. For one, the problem may be addressed directly by excluding or repositioning such mixed-link nodes or some of their neighbors. Another alternative is to employ wireless transceivers offering better suited transmission power control options. Finally, in some scenarios compromising the quality of the layer separation a bit by lowering the required signal strength spread at each receiver may be feasible, especially if only few or unimportant nodes are affected. Latter may lead to occasional inadvertent inversion of packet priorities and true destructive packet collisions, which some applications may be able to tolerate.

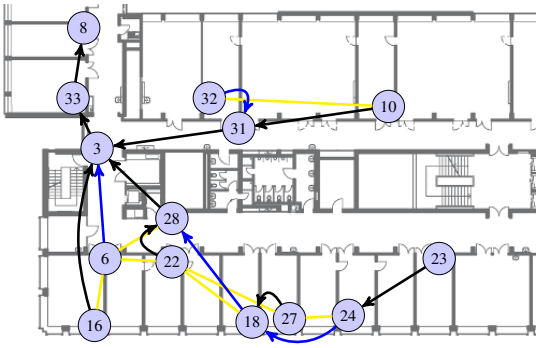

Figure 5. A part of the FlockLab testbed which we used to conduct our experiments on. An example of a convergecast tree with node 8 as the root node is shown. Yellow edges indicate links not part of the tree. To avoid collisions within the convergecast layer, no two sibling branches connected by links may execute simultaneously. Hence, every node, once it is woken, first queries all its black edge children in parallel, and then, in a second step, its blue edge children.

If one wishes to have multiple protocols have the same priority and be entitled to equal share of the medium, it is not advisable to assign both protocols the same layer of receive powers. Since packets of the same strength arriving at a node will not trigger the capture effect but instead lead to destructive packet collisions, none of the packets would be decoded correctly. Instead, we recommend having a separate layer for each protocol, but rotating through the protocol assignments for the layers on a slot number basis. I.e., for two protocols, simply swap their layer assignments every c slots. We suggest choosing the value of c to be around 50 to 100 to avoid each protocol suffering very frequent packet loss when both protocols are under load.

5 Example Application

To verify and measure the effectiveness of our method on real world wireless sensor networks, we chose an example application highlighting the supposed benefits of our method and implemented it on the wireless testbed FlockLab [14] which spans the floor of an office building (see Figure 5).

We consider the scenario of fire detectors covering the rooms of a building with the goal of reporting the outbreak of a fire as quickly as possible to a base station, or *root node*, which to the network is just a normal node with a special role. The root node has a wired connection to the building facilities and can escalate the alarm if it is informed of a fire by one of its neighbors. In its wireless capability it matches a regular node and as such it can only hear a local neighborhood of nodes, necessitating the propagation of a fire alarm over multiple hops.

Additionally, we require the liveness and proper functioning of all nodes to be regularly verified so that defunct nodes may be replaced in a timely manner. We regard these liveness tests to be of less urgency than the fire alarms and thus do not mind fire alarms having priority access to the wireless medium. Hence, in our model the liveness tests will be executed as a protocol in layer 1, while the fire alarm propagation will take place as a protocol on layer 2.

We model each fire detector as a TelosB sensor node extended with a smoke sensor, though for this experiment we only simulate a virtual smoke sensor to make alarm generation easier. The TelosB sensor node is mainly comprised of a MSP430 16-bit RISC microcontroller and a CC2420 wireless transceiver [24]. The CC2420 supports the IEEE 802.15.4 ZigBee wireless standard [7] and is capable of either sending or receiving on a single frequency at a time. It uses a preamble for detecting the start of packets and offers output power control, but does not provide any more advanced features such as message in message or decoding more than one transmission at a time. For all intents and purposes it fulfills the ideal of a "standard" low-power wireless interface as was referred to in Sections 3 and 4.

5.1 Layer 1

We design the layer 1 protocol as a parallelized convergecast on a tree overlaid onto the network connectivity graph. Every node knows of its parent and its children in the tree as well as the adjacency relationships between its child branches. The root node repeatedly initiates convergecasts and will, whenever a node is indicated as missing or defunct by the output of the convergecast, generate an appropriate alert for that node's replacement. This layer will also resynchronize a node's clock whenever it receives a message from its parent in the tree. This ensures the packet transmission synchronization requirements for achieving the capture effect hold over the time of the deployment.

The states each node goes through as it participates in the convergecast are depicted in Figure 6. The root node begins a convergecast by waking up itself and skipping to q_{wake} (since it has no parent). Other nodes start in q_{sleep} and wait for a request from their parent. After acknowledging the request ($q_{ackrequest}$) their parent will send some of its children proceed messages (q_{wake}), while the remaining children only receive acknowledgments and will have to wait at first ($q_{waitproceed}$). After "proceeding" a node wakes its children with a request of its own and then proceeds to q_{choose}.

Once a node has woken all its children it will start querying subsets of its children ($q_{sendproceed}$), such that the branches of no two chosen children are adjacent in the same subset. With every proceed message, every addressed child is assigned a recurring slot during which it may confirm the query (not pictured) and then later send a response, once it has gathered all the information from its branch. Once a subset of branches has completed, by each branch either sending a report or not responding for 3 subsequent queries, the process is repeated for another subset of branches. Finally, when a node has gathered all the information from its subtree it will wait for one of its designated response slots and report to its parent ($q_{respond}$).

Anytime a node needs to send messages to multiple of its children at the same time, it will combine these messages into one and send it at the highest power any of the links requires for layer 1. While this may strain the layering system in certain configurations, the effects were negligible in practice. Hence, we followed through with this approach for its efficiency and simplicity. An additional benefit is the compliance with the assumption that no node would ever send in

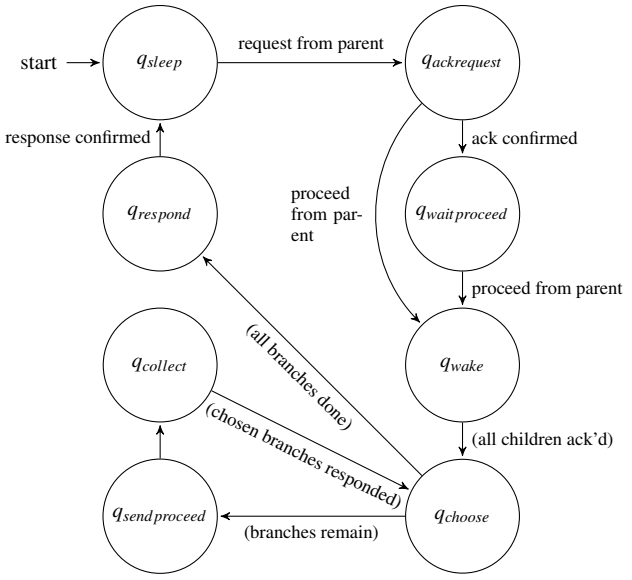

Figure 6. A high-level overview of the states each nodes traverses during the parallelized convergecast. Resends and root specific transitions were omitted for clarity.

two subsequent slots.

Further, every query and every report is explicitly acknowledged. If an acknowledgment is lost, the query or report will be resent after randomized exponentially growing backoff intervals until an acknowledgment is received or, in the case of a query, the recipient is pronounced dead after 3 attempts. Once a node is determined to be dead, it will be ignored for the remainder of the convergecast and the root node, upon receiving the aggregated node data, can notify building personnel to manually check on the potentially broken node. If a node retrying reports is queried for a new convergecast, it will snap out of its clearly erroneous state and participate as per usual in the new convergecast.

5.2 Layer 2

The fire alarm propagation is implemented in layer 2, i.e., a layer with a higher priority than the convergecasts described above. When an alarm event occurs it is propagated along a tree towards the root, a tree similar or even identical to the one used by layer 1. We simulate smoke alarm events occurring randomly and independently with a chance of 1% each slot. In a first test series, we trigger the event at one random node only (SA). In a separate test series we trigger multiple alarm events (MA) at different nodes simultaneously or with a few slots delay as might happen in the case of the outbreak of a real fire.

Multiple simultaneous alarms introduce an additional difficulty to the layer 2 protocol. As mentioned in Section 4.1, even the protocol with highest priority needs to deal with packet loss due to addressed nodes possibly not listening as they might be sending out a packet of their own on layer 1. The solution mentioned previously, to repeat all packets of the highest layer once in the subsequent slot, is not sufficient here due to the possible presence of multiple alarms, which in our setting all need to have the same priority (as we only have 2 layers available) and are hence expected to possibly collide destructively. Thus, we additionally implemented implicit and explicit acknowledgments for alarm propagation. When a node receives an alarm packet from a descendant in the tree, it forwards it in both of the next two slots and then listens. If it hears an ancestor in the tree propagate the alarm, it takes this as an implicit acknowledgment and calms down, i.e., no longer spreads the alarm. If it does not hear the alarm propagated, it repeats it another two times after a random exponentially growing backoff period, and listens again. A node which hears the same alarm again (after it had already propagated it) does not propagate it again. If the sender was a direct descendant, it sends back an explicit acknowledgment.

We also considered solving the collision of multiple alarms by tracking the received signal strength indicator (RSSI) when no packet is being received in a slot. We expected to see a signal strength similar to or stronger than that of a layer 2 packet, which would indicate that an alarm had occurred for certain, even if the exact data of the alarm was unavailable. This would allow us to pass on the existence of an alarm without incurring a latency penalty from the alarm collision. Unfortunately, experiments showed the RSSI to not be reliable enough of a measure for the presence of layer 2 packets, producing an unacceptably high rate of either false positives or false negatives.

5.3 Discovery

To determine the trees to be used by layers 1 and 2 as well as the different reception power levels for each layer at each node, we initially perform a "discovery" phase. The goal of this phase is to record the quality of each link in the network, compute the trees with the smallest possible height and then inform each node of its parent, its children and all the parameters required for operation as listed above.

While in our experiments we needed to perform this phase only a single time, in practice it would likely be desirable to repeat this phase every so often to deal with changes in the wireless environment, as, for instance, may easily be caused by the closing of doors or the increasing of a room's air temperature or humidity. Such repeated runs may for the most part be executed solely in layer 1 without impacting the operationally critical alarm propagation on layer 2, the potential exception being tests of links at higher sending powers. In our experience some links' qualities can drastically change every few seconds, while others are stable for days. Reasonably, one would not perform a complete discovery as often as every few seconds or minutes, but rather on the order of hours while testing known fluctuating links more frequently.

6 Test Results

We compare the performance of our method to a traditional approach, which does not incorporate the capture effect but for comparability's sake adheres to the time slotting. It will, however, still execute both of the protocols (convergecast and alarm propagation) and is aware of their relative priorities. Hence, it will prefer forwarding alarm packets over sending convergecast related packets, but will use the same transmission power for all packets. Additionally, we compare the latency of the alarms as well as the durations

Table 2. Experiment A: Percentages of successful alarms and convergecasts.

	Alarms	Convergecasts
Traditional (SA)	78%	98%
Layering (SA)	100%	88%
Traditional (MA)	59%	98%
Layering (MA)	79%	88%
Traditional (MA w/ acks)	100%	75%
Layering (MA w/ acks)	100%	88%

Table 3. Experiment B: Percentages of successful alarms and convergecasts.

	Alarms	Convergecasts
Traditional (SA)	79%	98%
Layering (SA)	98%	87%
Traditional (MA)	50%	97%
Layering (MA)	65%	53%
Traditional (MA w/ acks)	82%	66%
Layering (MA w/ acks)	98%	50%

of the convergecasts to the best physically possible values. Since these values usually are not obtainable without a dose of luck with regards to low-reliability long-range links, we do not expect these values to consistently be met.

In the first test series we consider the described algorithm without alarm acknowledgments (and thus without resends) in the scenario of single alarms (SA) only. We do, however, still repeat every alarm propagation packet in the subsequent slot to avoid losing it to a layer 1 packet being sent at the destination node. For the second and third test series we tested the algorithm with and without acknowledgements in the scenario of multiple alarms (MA). We present the results for two representative experiment runs with different topologies. Experiment A used 11 nodes and executed 462 convergecasts and 231 alarms. Experiment B used 14 nodes and focused on an increased alarm density by doubling the probability for an alarm to occur in each slot to 2%, executing 311 convergecasts and 528 alarms. For both experiments, the convergecasts and alarms were divided evenly among the 3 test series and 2 approaches. The experiments took approximately 50 minutes each.

Tables 2 and 3 list the portions of alarms and convergecasts which were successful in each setting. An alarm is successful if it did not get lost, i.e., reached the root node eventually. A convergecast is successful if it completed correctly and collected data from every single node. The tendency of the layered approach to promote alarms over convergecasts even more than the traditional approach is clearly visible: while our layering approach generally suffers from fewer successful convergecasts, it beats the successful alarm ratio of the traditional approach, reaching an almost certain alarm delivery. The difficulty of dealing with multiple alarms without acknowledgements is also apparent, as alarms inevitably get lost. Of special note is the fact that while the traditional approach reaches the same 100% alarm successes as us using acknowledgements in experiment A, it suffers a larger convergecast success penalty.

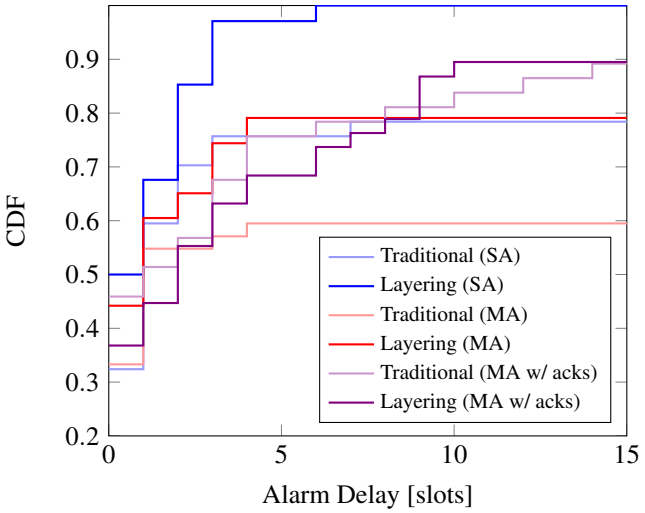

Figure 7. Experiment A: Distributions of alarm delay.

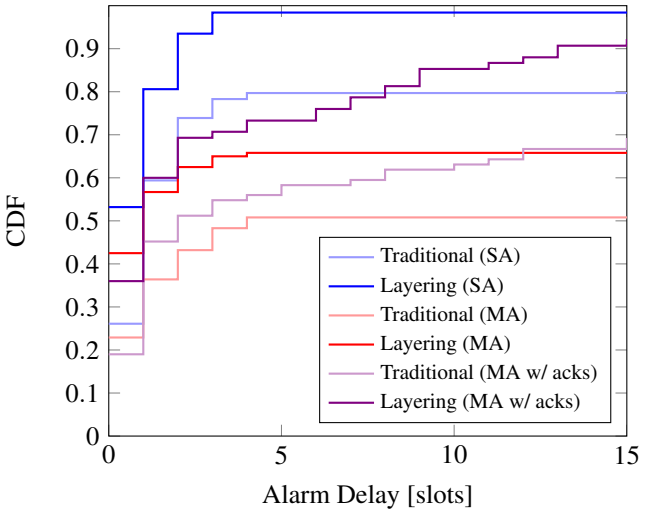

Figure 8. Experiment B: Distributions of alarm delay.

Figures 7 and 8 show the CDF (cumulative distribution function) for the alarm delay. The delay of an alarm is defined as the number of slots it takes to reach the root node minus the physical minimum number of slots required to traverse the multi-hop path from its origin to the root. This allows for a comparison between alarms originating at different nodes. We observe our approach beating the traditional one in each category, not least because it suffers fewer lost alarms, with the exception MA with acknowledgements in experiment A. For single alarms we achieve an alarm delay of 2 slots or less in 85% resp. 94% of cases, which is excellent considering the optimal reference alarm delay taking unreliable long-distance links into account. As was to be expected, overall the maximum delay experienced without acknowledgements is around 4 slots, while acknowledgements allows alarm reporting to be drawn out considerably.

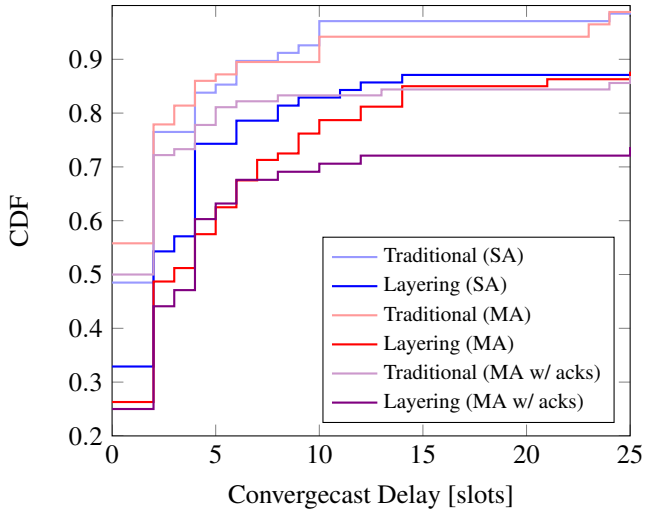

Figure 9. Experiment A: Distributions of convergecast delay.

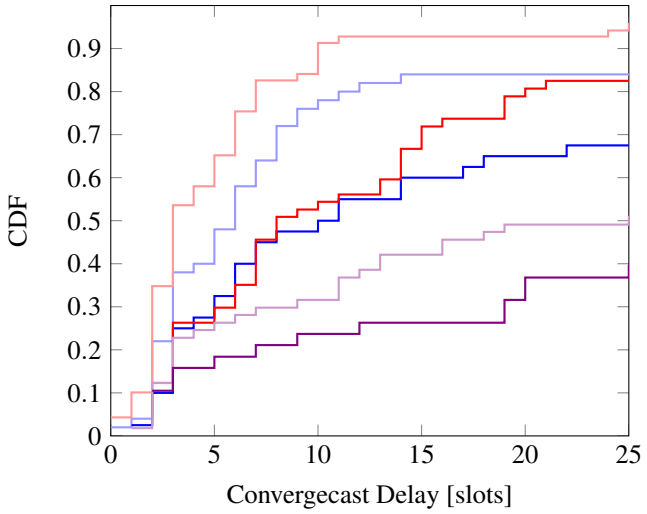

Figure 10. Experiment B: Distributions of convergecast delay.

Figures 9 and 10 shows a similar CDF for convergecast delays, defined as a convergecast's duration minus the minimum amount of slots our algorithm requires for the convergecast even if not a single packet was lost. The considerably worse performance of our approach here is due to it causing the layer 1 algorithm increased packet loss in order to support layer 2. Also of note is the general increase in delay for both approaches as more layer 2 traffic is introduced, both by adding acknowledgements and resends and by increasing the amount of alarms.

7 Conclusion and Future Work

We present a method to execute multiple protocols in parallel, giving each protocol the illusion of being the only one and having complete access to the network's resources. When a higher-priority protocol uses network resources, such as the ability of a node to send or receive a packet in a specific slot, lower-priority protocols experience this as packet loss, as sending priority or the capture effect drop lower-priority packets.

Further, in theory our method causes no overhead in terms of time slot use and latency. To confirm this, we implemented an example application with 2 protocols of different priorities and measured their performance. The results show very few packet losses and essentially optimal latency for the higher-priority protocol, while it causes additional losses to the lower-priority protocol compared to a traditional approach. In the scenario that both approaches can avoid loss of high-priority packets, our method does so while incurring less overhead to the lower-priority protocol.

Our technique does come with some downsides, most notably the removal of the individual protocols' ability to employ some options directly related to the physical layer. These options include power control, knowledge about corrupted packets, the use of the capture effect and the ability to not use time slotting. For many algorithms these features may be non-critical or even completely irrelevant. For others it may make this technique unsuitable or require significant changes. Further, the connectivity graph is likely to lose some edges on one or more layers whose target reception power ranges cannot be met by the respective senders' transmission power ranges. This especially affects the availability of long-range links on the lower-priority layers.

Our technique proves to be flexible concerning hardware and existing protocols, allowing it to be deployed without much additional overhead beyond implementing the discovery phase and basic slot logic. We have shown simple low-power hardware such as TelosB to be sufficient and pointed out how our technique is capable of masking the existance of competing protocols, allowing many protocols to simply be "dropped in" on one of the layers, possibly even on-the-fly. Some protocols may greatly benefit from being aware of the other protocols, for instance, by knowing when their packet was dropped already at the sender, or by even directly communicating with the higher layer protocols on the same node. We believe this work to nevertheless be a useful first step in the direction of exploring such multi-layer protocols.

Incidentally, our technique is also a feasible fairness provider in many situations. While in its basic implementation it will in fact allow any protocol to completely starve all protocols with a lower priority, as discussed in Section 4.2, by periodically reassigning the layers to different protocols, each protocol can be assigned an arbitrary share of the network resources when averaged over longer periods of time. However, our approach is likely not suitable for applications with a need for temporally more fine-grained QoS. This is under the assumption that under load each protocol is able to work more efficiently if it is the top active layer for its critical region for longer consecutive time intervals than for many smaller ones, separated by spurs of time with possibly 100% packet loss.

Power consumption was not examined as part of our tests. We believe this issue to be almost orthogonal to the prob-

lems discussed here, as duty cycling and rate of packets sent are barely affected by the proposed method. If nodes are unable to maintain synchronization through a sleep phase, care needs to be taken that they are resynchronized before attempting to send a packet intended to evoke the capture effect. On TelosB hardware, however, staying below the synchronization error threshold of 160 µs for longer periods of time is very easy due to its 32 kHz crystal oscillator (with a tick length of about 31 µs) being able to power-efficiently and accurately keep time even when the CPU is in sleep mode. While operation on higher-priority layers does require larger transmission powers, for many applications the proposed method may remain a desirable choice, especially if high-priority messages are more of an exception than a rule (as in our fire alarm example).

Promising future work includes applying this method to wireless hardware with a finer power control granularity, especially at the low power end. We expect such hardware to make it significantly easier to support a larger number of layers while having a clear separation of layers. In general, it is also worth investigating the capture effect parameters for a layering setup on other hardware, as hardware more amenable to exhibiting the capture effect may loosen the requirements on synchronization or layer separation.

If the lack of absolute network resource control for higher layers proves to be an issue (i.e., if lower layers are preventing reception of high layer packets by sending lower layer packets), various solutions may be worth exploring. For example, one could depart a little from strict slotting and have higher layers send their packets a few dozen symbol durations earlier, as to allow prevention of lower layer sending. Alternatively, a mechanism by which higher layers completely reserve some nodes for a period of time is imaginable and may be very effective depending on the application.

Acknowledgments

We would like to thank Roman Lim for his technical support and the anonymous reviewers for their valuable input.

8 References

[1] D. L. Ash. A comparison between ook/ask and fsk modulation techniques for radio links. Technical report, Technical report, RF Monolithics Inc, 1992.

[2] H. Chang, V. Misra, and D. Rubenstein. A general model and analysis of physical layer capture in 802.11 networks. In *INFOCOM*, 2006.

[3] A. Cidon, K. Nagaraj, S. Katti, and P. Viswanath. Flashback: Decoupled lightweight wireless control. *ACM SIGCOMM Computer Communication Review*, 42(4):223–234, 2012.

[4] D. H. Davis, S. Gronemeyer, et al. Performance of slotted aloha random access with delay capture and randomized time of arrival. *Communications, IEEE Transactions on*, 28(5):703–710, 1980.

[5] R. Flury and R. Wattenhofer. Slotted programming for sensor networks. In *Proceedings of the 9th ACM/IEEE International Conference on Information Processing in Sensor Networks*, pages 24–34. ACM, 2010.

[6] A. Ghosh, R. Ratasuk, W. Xiao, B. Classon, V. Nangia, R. Love, D. Schwent, and D. Wilson. Uplink control channel design for 3gpp lte. In *Personal, Indoor and Mobile Radio Communications, 2007. PIMRC 2007. IEEE 18th International Symposium on*, pages 1–5, Sept 2007.

[7] Institute of Electrical and Electronics Engineers. *IEEE Standard 802.15.4: Wireless Medium Access Control (MAC) and Physical Layer (PHY) Specifications for Low-Rate Wireless Personal Area Networks (LR-WPANs)*.

[8] N. Johansson, U. Körner, and P. Johansson. Performance evaluation of scheduling algorithms for bluetooth. In *Broadband communications*, pages 139–150. Springer, 2000.

[9] A. Kochut, A. Vasan, A. U. Shankar, and A. Agrawala. Sniffing out the correct physical layer capture model in 802.11b. In *Network Protocols, 2004. ICNP 2004. Proceedings of the 12th IEEE International Conference on*, pages 252–261. IEEE, 2004.

[10] J. Lee, W. Kim, S.-J. Lee, D. Jo, J. Ryu, T. Kwon, and Y. Choi. An experimental study on the capture effect in 802.11a networks. In *Proceedings of the second ACM international workshop on Wireless network testbeds, experimental evaluation and characterization*, pages 19–26. ACM, 2007.

[11] K. Leentvaar and J. H. Flint. The capture effect in fm receivers. *Communications, IEEE Transactions on*, 24(5):531–539, 1976.

[12] B. Li, L. Nie, C. Wu, H. Gonzalez, and C. Lu. Incorporating emergency alarms in reliable wireless process control. In *Proceedings of the ACM/IEEE Sixth International Conference on Cyber-Physical Systems*, pages 218–227. ACM, 2015.

[13] X. Li and Q.-A. Zeng. Performance analysis of the ieee 802.11 mac protocol over a wlan with capture effect. *Information and Media Technologies*, 1(1):679–685, 2006.

[14] R. Lim, F. Ferrari, M. Zimmerling, C. Walser, P. Sommer, and J. Beutel. Flocklab: A testbed for distributed, synchronized tracing and profiling of wireless embedded systems. In *Information Processing in Sensor Networks (IPSN), 2013 ACM/IEEE International Conference on*, pages 153–165. IEEE, 2013.

[15] R. Litjens, F. Roijers, J. Van den Berg, R. J. Boucherie, and M. Fleuren. Performance analysis of wireless lans: an integrated packet/flow level approach. *Teletraffic Science and Engineering*, 5:931–940, 2003.

[16] J. Lu and K. Whitehouse. *Flash flooding: Exploiting the capture effect for rapid flooding in wireless sensor networks*. IEEE, 2009.

[17] J. Manweiler, N. Santhapuri, S. Sen, R. R. Choudhury, S. Nelakuditi, and K. Munagala. Order matters: transmission reordering in wireless networks. *Networking, IEEE/ACM Transactions on*, 20(2):353–366, 2012.

[18] A. Nyandoro, L. Libman, and M. Hassan. Service differentiation using the capture effect in 802.11 wireless lans. *Wireless Communications, IEEE Transactions on*, 6(8):2961–2971, 2007.

[19] P. Patras, H. Qi, and D. Malone. Mitigating collisions through power-hopping to improve 802.11 performance. *Pervasive and Mobile Computing*, 11:41–55, 2014.

[20] J. Polastre, R. Szewczyk, and D. Culler. Telos: enabling ultra-low power wireless research. In *Information Processing in Sensor Networks, 2005. IPSN 2005. Fourth International Symposium on*, pages 364–369. IEEE, 2005.

[21] S. Ray, J. B. Carruthers, and D. Starobinski. Rts/cts-induced congestion in ad hoc wireless lans. In *Wireless Communications and Networking, 2003. WCNC 2003. 2003 IEEE*, volume 3, pages 1516–1521. IEEE, 2003.

[22] I. Rhee, A. Warrier, M. Aia, J. Min, and M. L. Sichitiu. Z-mac: a hybrid mac for wireless sensor networks. *IEEE/ACM Transactions on Networking (TON)*, 16(3):511–524, 2008.

[23] D. Son, B. Krishnamachari, and J. Heidemann. Experimental study of concurrent transmission in wireless sensor networks. In *Proceedings of the 4th international conference on Embedded networked sensor systems*, pages 237–250. ACM, 2006.

[24] Texas Instruments. *2.4 GHz IEEE 802.15.4 / ZigBee-ready RF Transceiver*. CC2420 Data Sheet.

[25] K. Whitehouse, A. Woo, F. Jiang, J. Polastre, and D. Culler. Exploiting the capture effect for collision detection and recovery. In *Proceedings of the 2nd IEEE workshop on Embedded Networked Sensors*, pages 45–52, 2005.

[26] K. Xu, M. Gerla, and S. Bae. How effective is the ieee 802.11 rts/cts handshake in ad hoc networks. In *Global Telecommunications Conference, 2002. GLOBECOM'02. IEEE*, volume 1, pages 72–76. IEEE, 2002.

[27] D. Yuan and M. Hollick. Let's talk together: Understanding concurrent transmission in wireless sensor networks. In *Local Computer Networks (LCN), 2013 IEEE 38th Conference on*, pages 219–227. IEEE, 2013.