A Channelization Protocol for Multi-hop Wireless Sensor Networks using Frequency Division Multiplexing

Biswajit Mazumder, Vijayakrishna Venugopalan, Jason O. Hallstrom
School of Computing
Clemson University
Clemson, SC 29634
{bmazumd, vvenugo, jasonoh}@cs.clemson.edu

ABSTRACT
We present a channelization protocol based on frequency division multiplexing to enable fast, reliable, multi-hop data transmission with minimal protocol overhead in wireless sensor networks. The protocol enables data transmission to occur at a high rate by eliminating intra-path collisions along a multi-hop routing path. This is achieved by staggering communication frequencies such that no two adjacent nodes transmit data at the same radio frequency. We describe the design and implementation of the protocol and present extensive experimental results to validate its efficacy.

Keywords
Frequency division multiplexing, multi-hop networking, sensor networks, TinyOS

1. INTRODUCTION
Many wireless sensor network (WSN) applications, such as disaster management, volcanic activity monitoring, and structural health monitoring [11, 12, 13] require high data collection rates along with high network reliability. In many cases, the data must be transferred over large distances. Single hop wireless transmissions are generally not feasible over long distances using low power radios. Instead, short-range, hop-by-hop transmissions are typically used. This also overcomes some of the signal propagation difficulties in long-distance wireless transmissions [2].

Data propagation over multiple hops involves transmission of data via intermediate nodes, until the message reaches its final destination. Multi-hop routing protocols tend to use flooding to achieve this. Each data packet is broadcast into the network. The transport layer uses retransmissions to achieve reliable data transmission. An important issue in such a protocol is that more often than not, several nodes in a given vicinity attempt to broadcast a message simultaneously. This results in radio interference at the MAC layer, perceived as collisions. When a collision is detected at a potential receiver, the message is rejected; the result is that none of the affected messages are delivered [2,3].

A number of transport layer collision avoidance strategies have been suggested to increase the performance of multi-hop communication. One solution is to use time division multiplexing (TDM). This involves the implementation of a random back-off mechanism where a data transmission from a node may be deferred for a short period to avoid collision with neighboring nodes [2]. But this mechanism compounds the back-off delay throughout the network as the number of hops increases, and hence inhibits high data rates.

Spatial multiplexing is another solution to the collision problem, as proposed in the Flush protocol [5]. Flush implements spatial multiplexing in a multi-hop wireless network such that not more than one node out of any three consecutive nodes in a linear topology is sending data at any instant. This ensures that data sent to a next-hop node will not result in a collision. The process is replicated at the next node and so on until the data has been received at a sink node. Flush requires a fixed percentage of nodes in the network to be inactive at any given time, thus limiting the maximum achievable data rate.

Our primary goal in this paper is to maximize multi-hop data rates. We describe a mechanism to ensure continuous transmission of data with minimal packet collisions. We extend the idea introduced in [5] by reusing multiple frequencies instead of reusing the space dimension and develop a channelization protocol based on frequency division multiplexing (CHAMP). We also try to achieve data rates that are independent of the number of hops in the network.

Multi-path, multi-hop network protocols enable higher data rates [6] than single path protocols. A single large flow can be divided into several sub-flows, and each sub-flow can be transmitted on a different path. Hence, we also outline a multi-path network protocol, which might later be implemented using CHAMP as its foundation. This would further increase the data rate performance of CHAMP.

Paper Organization. Section 2 surveys related work in the area. Section 3 describes the CHAMP protocol. Section 4 presents the implementation of CHAMP. Section 5 evaluates the performance and scalability of the protocol. Section 6 describes a multi-path routing protocol based on CHAMP. Section 7 presents concluding remarks.

2. RELATED WORK
Optimal bandwidth utilization is one of the foremost challenges facing WSN transport protocol designers. It is imperative that WSN protocols be scalable, while offering high
reliability and high data rates [4]. The challenge is in contending with the broadcast nature of the channels and the unreliable nature of the wireless medium itself.

Wireless networks have only one mode of communication—broadcast. Although packets may be filtered at higher layers to simulate unicast or multicast, at the physical layer, all packets can be sensed by any wireless receiver listening on the appropriate channel. Consider a multi-hop network in the absence of collision avoidance strategies. At any hop, a packet intended for a particular node can cause a collision at other nodes in the same neighborhood. This is illustrated in Figure 1. Node B is within the radio range of nodes A and C. In this scenario, while A is transmitting data to B, if C attempts to transmit data to D, the transmission will cause a collision at B, resulting in the loss of data being sent by A.

Figure 1: Collision caused at node B when both node A and node C transmit data.

Time division multiplexing (TDM) is a mechanism used in networks that require multiple nodes to transmit data over a shared medium. One way to achieve a TDM-like implementation in the context of WSNs is to use a random back-off mechanism to ensure that no two transmitters in a given vicinity send data at the same time [2]. However, one of the major disadvantages of the protocol discussed in [2] is that two different packets transmitted after the same back-off period will still result in a collision. Another disadvantage is that the “dead time” during back-off slots limits the potential bandwidth of the channel. Hence, for better bandwidth utilization, time synchronization becomes a necessity in protocols implementing back-off timers. Frequency Division Multiplexing (FDM) is the traditional way of separating radio signals among different transmitters. FDM is not vulnerable to timing problems since a predetermined frequency band is allocated for the entire period of communication.

The Flush protocol [5] describes a spatial multiplexing technique to control data transmission across a multi-hop network. Flush addresses the intra-path interference problem that occurs in scenarios like the one illustrated in Figure 1. This is achieved by synchronizing transmissions at every fourth node. Thus, only one third of the network is active at any given instant. This is illustrated in Figure 2, where the dark circles denote nodes actively involved in transmission, and the light circles denote nodes actively involved in reception (or in an idle state). The arrows indicate the general direction of data flow.

While Flush provides a reasonable approach to data transmission, it forces some nodes to be wastefully inactive. Specifically, Flush keeps 1/3 of the network inactive at every instant, limiting the maximum achievable throughput. We attempt to address this issue in CHAMP by allowing all neighboring nodes in a given vicinity to actively participate in data communication by operating at different frequencies.

3. CHAMP

The CHAMP protocol involves three types of nodes: a source, a sink, and multiple relay nodes. The underlying routing protocol is responsible for establishing a path from the source node to the sink node. The source node and the relay nodes send data in a persistent manner: Each node transmits each packet until an acknowledgement is received from the next-hop node. This is the key to reliable communication in CHAMP; no packet is ever lost. But there is a cost: The number of retransmissions is often high. End-to-end acknowledgements and selective negative acknowledgements (with retransmission) are alternative mechanisms for achieving reliability. Both, however, require packet buffering to handle out-of-order packets. Since nodes are resource-constrained, this level of packet buffering is not feasible. In the absence of packet buffering, out-of-order packets would have to be discarded and retransmitted.

3.1 Frequency Division Multiplexing

The most important aspect of the protocol design is channelization using frequency division multiplexing. We use the available frequency bands to transfer data simultaneously, in a shared medium, without collisions. We assign two channels to every node, a Tx channel and an Rx channel, separated by a configurable guard band of 15 MHz. The frequency at which a node sends data is equal to the receive frequency of its successor in the transmission chain.

To make the presentation more concrete, consider the CC2420 radio from Texas Instruments [8], a popular chipset for sensor network platforms. The CC2420 operates on 16 channels, numbered sequentially from 11 to 26. The correspondence between channel number and frequency is given by the following formula:

\[ F_c = 2405 + 5(k - 11) \text{MHz} \]

where, \( k = 11, 12, 13, \ldots, 26 \).

The protocol is responsible for assigning transmission and reception frequencies to each node based on its hop distance from the sink. When a node ascertains its position in the network, it computes its transmission and reception channels.
using the following formulas:

\[ TxChannel = 24 - (((d - 1) \mod 5) \times 3) \]  
(1)

\[ RxChannel = 24 - (((d) \mod 5) \times 3) \]  
(2)

where \( d \) denotes the node’s position in the relay chain, measured from the sink node. Accordingly, the transmission and reception channel variables can take on 5 values: 24, 21, 18, 15 and 12. Note that for \( d = 0 \) (i.e. the sink node) there is no transmission channel.

\[ X \]
\[ W \]
\[ Y \]
\[ X \]
\[ Z \]
\[ Y \]
\[ P \]
\[ N \]
\[ S \]
\[ W \]
\[ X \]
\[ Y \]

Figure 3: Retransmissions in CHAMP.

Given that the sending and receiving channels are different for each node, each device must periodically switch between the two channels. Since any 5 consecutive nodes in the relay chain do not transmit data at the same frequency, the collision problem is avoided. However, the approach introduces the possibility of packet delay at each relay point. Consider the scenario illustrated in Figure 3, where a node \( N \) is transmitting a packet on its transmission channel \( Y \). (The inactive channels are depicted by the dark circles.) If a predecessor \( P \) attempts to send a packet on \( N \)'s reception channel \( X \), the packet will not be received or acknowledged. \( P \) will receive an acknowledgement only if \( N \) is in the receiving mode. As a result, \( P \) will continue to retransmit the packet until it is received successfully. More precisely, the packet will be received when \( N \) finishes transmitting data to its successor \( S \) and switches back to the receiving state.

4. IMPLEMENTATION

To evaluate the performance of CHAMP, we implemented the algorithm for TinyOS [1]. We tested our implementation using the CC2420 radio chipset on the Telos platform [7]. The CC2420 is an IEEE 802.15.4 compliant radio, which provides a digital received signal strength indicator (RSSI) that may be read at any time to determine the amount of radio activity on the current channel. The radio transmits at a maximum rate of 250 kbps, but using TinyOS, the maximum achievable transmission rate in a single hop network is about 35 kbps. This is mainly due to the MAC backoff times introduced by the operating system, coupled with the delay introduced when switching from Rx to Tx mode, and vice versa, after each transmission [8]. We could achieve higher throughput by changing the underlying OS and MAC layers, but that is beyond the scope of this paper.

4.1 Source Node

The source node is responsible for transmitting data packets at the highest rate possible. In our implementation, the source forwards 26-byte packets to its next node. We use 26-byte packets since, based on empirical observation, this is the maximum packet size that can be transmitted by TinyOS. In each case, the packet payload contains numeric patterns, which are later matched for correctness at the sink node, in addition to the normal CRC checks at the link level. At boot time, the source node initializes the radio. Next, it attempts to join a spanning tree rooted at the sink node using a standard spanning tree protocol. Next, it assigns itself a transmission (Tx) frequency based on its distance from the sink, measured in hops. The Tx frequency is established using formula (1).

4.2 Sink Node

The sink node is responsible for packet reception and data transmission to a serial-attached PC. At boot time, the sink node initializes the radio and sets its reception (Rx) frequency to 2470 Mhz (channel 24). Next, it initiates the spanning tree routing protocol. When the routing tree has been established, the sink begins receiving data packets through the radio. Each received packet is forwarded to the attached PC using a standard serial connection. If the sink node receives data faster than it can relay data over the serial interface, packets are buffered; buffer size can be adjusted programmatically.

The serial connection transmits data at a rate of 115.2kbps. The maximum achievable data rate with the chipcon CC2420 radio with 29-byte payloads is 23,200bps [5]. This suggests that we typically will not see a bottleneck at the radio-to-serial interface.

4.3 Relay Node

Each relay node is responsible for receiving data from a predecessor node and forwarding this data to a successor node. At boot time, each relay node initializes its radio and attempts to establish a successor, and thus participate in establishing path to the sink node using the spanning
tree routing layer. Next, the node calculates its Tx and Rx channel frequencies based on its hop distance from the sink.

The state transition diagram shown in Figure 4 illustrates the process used by relay nodes in switching among their constituent states. Each relay node attempts to forward each packet to the next node the instant it is received. On packet reception, a relay node sets its operating frequency to its Tx channel and moves to the send state. On successful completion of a transmission, the node reselects its receive frequency and waits for the next packet. On an unsuccessful transmission, the node moves back to the send state and retransmits.

Each relay uses hardware-level acknowledgements, implemented within the link layer of the CC2420.

5. EVALUATION

We now revisit our protocol design goals, so as to better appreciate the results of our evaluation. CHAMP aims to achieve high data rates, without collisions, reliably, over multiple hops with minimal protocol overhead. We evaluate our success in achieving these goals by performing a series of experiments to measure the performance of CHAMP using Telos [7] devices.

5.1 Experimental Setup

We use the minimal spanning tree protocol to form a standard collection tree with the sink node located at the root. The protocol requires that each participant broadcast specialized management packets to assist in constructing the tree. Once the tree has stabilized, the management features of the protocol are disabled before sending experimental data over the network. This ensures that the measured data rates are not affected by protocol management traffic.

We use a radio frequency (RF) power of 1 (-40.977 dBm), which limits the geographic scale of our network. We run each set of experiments three times for a duration of one minute and calculate the following metrics to evaluate the performance of our protocol:

- **Overall Throughput**: The number of data packets (or bytes) received at the sink divided by the total transfer time.
- **Overall Retransmissions**: The total number of packet retransmissions across the network during a given period.

We began by attempting to identify the minimum distance at which two communicating nodes would not interfere, as measured by the packet reception rate (PRR) between a transmitter and a receiver. In principle, the resulting distance, \( d \), could then be used as the inter-node distance in our linear test topology. However, to ensure that the PRR remains high (i.e., approximately 1), we reduce this distance by one-half in our experimental setup. This ensures high-quality wireless links, while also guaranteeing interference freedom. The latter point is explained by noting that our experimental topology relies on 5 different channels. Hence, nodes communicating on the same channel are separated by a distance of 2.5\( d \) – well-beyond the interference threshold.

Given our use of the packet reception rate between two nodes as a proxy for measuring interference, it was important to control for factors that degrade the packet reception rate externally. More specifically, it was important to conduct the measurements using a channel that offered the lowest noise floor. Since the received signal strength indicator (RSSI) value of a packet is directly related to the noise floor in a wireless medium, we first developed an RSSI scanner. The application scans channels 11 to 24 of the CC2420 radio and samples RSSI values at a rate of 1 Khz, streaming the samples to a serial-attached PC. The PC application then computes the channel with the lowest noise floor and constructs a histogram depicting the collected data. Figure 5a shows the output of the RSSI scanner near the outdoor testbed region. Figure 5b is a sample output of the RSSI scanner in the presence of high microwave activity in an indoor environment. Figure 5a depicts minimal wireless activity, whereas Figure 5b exhibits high levels of wireless noise. Using this tool, we were able to determine that channel 11 had the lowest noise floor in the outdoor testbed.

Using channel 11, we found that at a distance of 4ft and an RF power level of 1, nodes on the same channel no longer interfere. We used half this distance (2ft) for our testbed setup. Hence, we deployed a 21-node (20 hop) outdoor testbed, arranged in a linear fashion, with a distance of 2ft between each node, in an open area devoid of any wireless activity, as shown in Figure 6.
5.2 Performance

We now examine the throughput achieved using CHAMP. We sent 26-byte data payloads across the network for a period of one minute for each of three runs. In each case, the hop count was varied from 1 to 20 hops. The TinyOS message structure adds a 10-byte header to the payload, for a message size of 36 bytes. We measured the overall data rates, actual payload data rates, and total number of retransmissions in the network per minute. Figure 7 summarizes the results of these trials.

Figure 7: Average throughput over multiple hops.

The results reveal that beyond two hops, CHAMP maintains a steady overall data rate of approximately 8500 bps. Over a single hop, CHAMP achieves a data rate of 24720 bps. We see that the data rate CHAMP achieves for 20 hops is between one-third and one-half of the single hop data rate. Since single-hop data transfer rates are the limiting factor for multi-hop rates, we argue that we can achieve better multi-hop performance with CHAMP if we can increase the data rate achieved by our implementation in a single-hop scenario. (We will soon return to this point).

We can see from Figure 7 that the data rate of our protocol drops substantially in moving from a single-hop network to a two-hop network. This can be explained by the absence of a relay node in the single-hop network. If a relay node takes \( t_1 \) seconds to receive \( b \) bits of data and \( t_2 \) seconds to transmit them, the effective throughput becomes \( b / (t_1 + t_2) \) bps, which is approximately half the maximum data rate achievable from a single-hop network. Also, since the relay node must shift continuously between the Tx and Rx channels, a significant amount of time is spent in the initial MAC back-off period. All these factors contribute to the drop in data rates beyond the two-hop network. Packet copying is another major bottleneck in high-throughput packet forwarding networks [9]. This, along with multiple other OS delays, cause the substantial drop in data rate. One way to remedy this decline would be to reduce the MAC delays and introduce zero-copy techniques during packet forwarding.

Figure 8 illustrates an interesting phenomenon: an approximately linear growth in the number of retransmissions as hop count is increased. This is due to the fact that each relay node introduces a time period in which it retransmits data because its successor is busy sending data. Retransmissions should not, however, be confused with collisions. CHAMP does not introduce any data loss due to collisions.

Figure 8: Total network retransmissions over multiple hops.

A network subject to congestion (and retransmissions) can be stabilized by adding exponential damping (i.e. an exponential back-off timer) to its source and relay nodes [10]. Hence, we suggest that adding back-off timers in CHAMP would result in a reduction in the number of retransmissions and ensure a congestion free protocol.

It is also worth noting that CHAMP introduces minimal protocol overhead to achieve reliable and stable data rates. We use a 1-byte retransmission counter to measure the total number of retransmissions in the network, which is included in the data packet size of 26 bytes. Recall that the network packet size is 36 bytes. Thus the payload throughput yield is 72% of all data sent. This means CHAMP has an effective average bandwidth of 17854 bps over a single hop.

5.3 Scalability

As illustrated in Figure 7, the protocol offers a stable data rate for all cases beyond the single-hop case. Protocol performance is unaffected by network size. We have also used 15 Mhz guard bands, which can be further reduced to accommodate additional channels. Increasing the number of available channels would enable nodes using the same frequencies to be spaced at larger distances.

6. FUTURE WORK

CHAMP can be further improved to obtain higher data transmission rates by integrating multi-path routing mechanisms. Section 6.1 describes how such a protocol might work. In order to achieve higher data rates and reliability in these protocols, the network deployment needs to be structured (i.e. pre-planned). There are many sensor network applications which involve fixed sensor network topologies, along with high data collection rate requirements [11, 12, 13]. The following strategy used in tandem with CHAMP could be advantageous in such scenarios.

6.1 Multi-path Spatial Multiplexing

The frequency division multiple access scheme introduces a significant bottleneck in CHAMP. At any given node, the Rx channel is unavailable while the node is in the Tx state. This results in un-ACKed packets and a large number of retransmissions. One way to overcome this bottleneck is by introducing multi-path routing. We describe a multi-path
routing mechanism based on CHAMP to implement this.

Figure 9: A multi-hop, frequency-based channelization mechanism.

Figure 9 illustrates a sample topology comprising two non-interfering paths from the source to the sink. Data is sent over each path using CHAMP. The source node alternates between the two successor relay nodes while transmitting data. A packet buffer may or may not be implemented at each of the relay nodes. The source node transmits data packets to one of the two relay nodes until the node’s buffer (buffer size is one in unbuffered mode) is full. When the relay node’s buffer is full, it moves into the Tx state, where it forwards all collected data to its successor node. On emptying its buffer, the node shifts back to the receive mode to begin receiving packets from the source node. When the relay node is in Tx mode, the source node retransmits the next packet a few times, but no longer receives ACKs. This indicates that the relay node has shifted to the Tx mode. Hence, the source shifts its target successor to the second relay node and repeats the Tx process.

Since data packets may arrive out of order in such a protocol, sequence numbers are required for all packets, as well as a resequencing buffer at the sink node. In a Utopian environment, if the buffer sizes and data transfer rates are constant across the network, the data flow will be equally distributed between the two paths. If each path attains its maximum achievable data rate, we would expect a data rate that is close to twice the rate achieved by the single-path protocol. However, in a practical implementation, the gains will be less dramatic.

7. CONCLUSION

A channelization strategy based on frequency division multiplexing works well for wireless sensor networks where high data rates and reliability are required. We designed, prototyped, and evaluated such a protocol. We have shown that the data rates achieved by our protocol are independent of the number of hops in a network. However, the challenge is to be able to achieve an even higher fixed data rate, while retaining this hop independence. We have suggested a novel approach, using which, we believe the achievable data rate can be further improved. Hence, we have demonstrated that using frequency division multiplexing based strategies for bulk data transmission is feasible and may form the basis of a reliable and scalable transport protocol.

Acknowledgments. This work was supported in part by the National Science Foundation (CNS-0745846).

8. REFERENCES