Using multiple communication channels for efficient data dissemination in wireless sensor networks

Robert Simon, Leijun Huang, Emerson Farrugia and Sanjeev Setia
Department of Computer Science
George Mason University
Fairfax, VA USA 22030
{simon, lhuang2, efarrugi, setia}@cs.gmu.edu

***Abstract*— This paper presents McTorrent and McSynch, two multichannel sensor network protocols for data dissemination. Both protocols are designed to take advantage of the spatial multiplexing properties of the half-duplex radio transceivers available on the current generation of sensor nodes. McTorrent is used for reliable end-to-end dissemination of a large data object. Compared to existing protocols, we show that McTorrent significantly reduces the amount of time required to propagate a large data object throughout a sensor network. McSynch is used to achieve data object synchronization within a local cluster of nodes. By using a scheduled channel access approach and an appropriate number of transmission channels, McSynch can significantly reduce the amount of time required to update a local cluster. We also describe our experiences implementing a multichannel system, and report on lessons learned for channel and frequency settings.**

I. Introduction

An essential feature of most sensor networking protocols is the ability to minimize and carefully manage communication in order to achieve energy savings. One attractive approach to the design of sensor network protocols and algorithms is the use of multiple communication channels. Such an approach can achieve both a higher level of effective throughput and better energy savings by using multiple communication paths, since transmissions on different channels do not interfere with each other. This paper explores the use of multiple channels in sensor network protocols for data dissemination.

Multichannel communication radios have been in existence for several decades. However, most multichannel network protocols that have been proposed and implemented to date are designed for mobile cellular telephony or for more traditional mobile ad hoc networking architectures such as might be found in an 802.11 environment [1]. The current generation of wireless sensor motes provides multiple channels available for use. For instance, the Crossbow MICA2 MPR400 comes equipped with a Chipcon CC1000 radio that has 240 channels [2]. To our knowledge, this work is one of the first to explore the use of multiple channels in sensor network protocols.

In order to effectively exploit the multichannel capability of a sensor node, it is necessary to define a new class of networking protocols that can take advantage of multiple channels. This problem is exacerbated by the fact that many currently available radios for sensor nodes are equipped with one half-duplex transceiver. Although the transceiver can be programmed on the fly to switch channels, each node can only transmit or listen on one channel at a time. The practical impact of this hardware feature is that multichannel networking protocols must be carefully designed to efficiently coordinate channel sending and receiving assignments. Thus, it is necessary for senders and receivers to coordinate both which channel they are assigned to *and* whether they are in a sending or receiving state.

The traditional approach to network protocol stack design is to completely decouple a multichannel MAC layer from upper level protocols. However, our approach is to use upper layer protocols to coordinate channel and sending and receiving assignments. Cross layer techniques are typical of much recent work in sensor networking [3]. The motivation for this arises from the resource constrained nature of sensor nodes. By exposing channel and transceiver mode selection to the higher layers, we achieve better communication performance and avoid the need to duplicate data and control code in multiple layers.

We present and discuss two protocols. The first is *McTorrent*, a protocol for multihop reliable data dissemination. The McTorrent protocol can be used for distributing a large data object to all the nodes in the network, e.g., distributing a new code image to sensor nodes for network reprogramming [4], [5], [6]. It uses a contention-based approach to channel access, and does not require time synchronization. As compared to a single channel approach, McTorrent significantly reduces the total latency for reliably disseminating a large data object from a sink. Our second protocol is *McSynch*, a cluster-level protocol for localized data synchronization and coordination. McSynch uses a distributed scheduling approach for channel access, and is designed for situations where nodes might be asleep for extended periods of time and when new sensors are added to the network. We show that by using McSynch and the appropriate number of channels it is possible to substantially reduce the time required to achieve cluster-wide synchronization. Note that both protocols use the prefix *Mc*, for multichannel.

We also report on the implementation and analysis of multichannel communication within a sensor node testbed. Our results indicate that, although it is necessary to pay careful

0-7803-9466-6/05/$20.00 ©2005 IEEE

attention to the control of frequency settings and channel separation, the current generation of sensor nodes can readily support multichannel communication.

II. Related Work

This section describes related work on the use of multichannel wireless networking, along with some recent reports on data dissemination in a sensor network.

A. Multichannel wireless systems and sensor networks

There are many papers that have dealt with the benefits of using multiple channels as a means to achieve higher throughput, often by also dealing with hidden terminal problems. One way of characterizing multichannel wireless networking research is by the number of transceivers and the assumed data link protocol. Examples of protocols that require multiple transceivers in a CSMA environment include Nasipuri and Das [7]. The works of Wu et al. and Jain et al. also discuss protocols that dynamically assign channels, although they differ as to how the channels are assigned [8][9]. Examples of single transceiver approaches are presented in [10] and [11]. Both use a handshaking approach in a frequency-hopping spread spectrum system (FHSS) to improve throughput by avoiding hidden terminal problems. The drawback to these efforts is that they can only work in an FHSS environment. So and Vaidya describe a MAC layer protocol that also requires only one transceiver per host, and solves the hidden terminal problem using temporal synchronization [12]. There has also been some work done in data striping using multiple channels, which requires a separate radio interface for each channel [13].

All of the above research focuses on the 802.11 protocol, or suggests enhancements to that protocol. In contrast, our paper deals with multichannel sensor networking. The major issues in this environment are effective cross-layer and application-aware design, along with communication management that minimizes the number of messages and collisions.

One critical issue is how well the current generation of sensor radios will actually perform in a real environment. For instance, the impact of radio irregularity on packet reception has extensively been studied in single channel sensor environments [14]. To our knowledge, the work here represents the first reported results on the use of multichannel radios from a real testbed.

B. Data dissemination

Protocols for data dissemination in sensor networks can be divided into two categories based on the size of the data objects that are being disseminated. The first category of protocols is designed to exchange a relatively small amount of data among neighboring nodes, whereas the second class of protocols is intended for disseminating very large data objects.

Protocols in the first category include the SPIN family of data dissemination protocols [15]. The SPIN protocols take advantage of the broadcast nature of the wireless medium, and use various communication suppression mechanisms to reduce latency. SPIN uses a three-phase (advertisement-request-data) handshaking protocol between nodes to disseminate data. The Trickle [16] protocol proposes mechanisms for dynamic broadcast rate adjustment with the goal of quickly propagating updates while reducing communication between nodes when there are no updates. Unlike SPIN and Trickle, our protocol, McSynch, takes advantage of multiple channels.

The second category includes several protocols such as MOAP [4], Deluge [5], and MNP [6], that have been proposed for multihop network reprogramming in a sensor network. Network reprogramming is an important application of reliable data dissemination protocols for sensor networks. All these protocols share some basic characteristics. First, these protocols are used for entire code image delivery as opposed to difference-based application adjustment [17]. Second, these protocols extend the three-phase handshaking protocol proposed in SPIN-BL [15] for handling large data objects. Third, the protocols all borrow ideas such as the use of selective NACKs and hop-by-hop error recovery from prior work in reliable data transfer protocols[18]. Deluge and MNP differ from MOAP in that a node does not need to receive the entire code image before it can rebroadcast it. By breaking the code image up into pages, and allowing pipelined page delivery, Deluge and MNP take advantage of spatial multiplexing to reduce the latency of network reprogramming. Deluge leverages the Trickle protocol [16] to limit transmissions between neighboring nodes. Our protocol for large object dissemination, McTorrent, resembles Deluge and MNP in its design with the crucial difference that McTorrent exploits multiple communication channels for data dissemination.

III. Multichannel Sensor Networking

Effective use of multichannel sensor networking is complicated, due to the need for channel selection and specification. Here we briefly discuss some multichannel architectural considerations and present our approach to reliable data dissemination.

A. Architectural Considerations

There are several current examples of low-cost multichannel radio technology. One typical example of current multichannel technology is the CC1000 radio, manufactured by Chipcon. The CC1000 radio on a MICA2 MPR400 node is set to operate in the frequency region between 902MHz and 928MHz. The radio cannot tune to an arbitrary frequency. Rather, it relies on a digital frequency synthesizer to select from 240 discrete channels at an approximate channel spacing of 100KHz. However, as shown experimentally in Section VI, this channel spacing is insufficient to prevent adjacent channel interference.

We have developed an API that extends basic TinyOS functionality for accessing different channels. TinyOS provides a control interface to the CC1000 radio. The interface is used to alter transmit power, select a channel, or set the radio to either transmit mode or receive mode. The channel selection function computes the discrete channel having the closest base frequency to a desired value, allowing a node to theoretically change its channel on demand. At the time of this writing,

inconsistencies between the code and the hardware still require the radio to be turned off during channel selection, which adds approximately 30ms of downtime.

We expect that the above API is generalizable to multiple radio types. One important issue is whether or not the data link can use time synchronization to achieve transmission collision freedom, using a TDMA-like approach. To account for this case the API allows the link-level scheduling function to specify a parameter indicating which time slot to use.

B. Data dissemination

A central focus of our work is to describe multichannel protocols for reliable data dissemination of some data object *OBJ*. The size of this data object may be large, as in the case of new code for node reprogramming, or small, as in the case of maintaining shared state for local decision making. The data object is of size S_{obj}, and is divided into fixed size packets of size S_{pkt}. In order to facilitate efficient communication control and data management, the size of the packets is the same as the size of the data link transfer unit.

By dividing *OBJ* into a series of packets, it is possible for each node to represent the complete object as a bit vector $BV\langle OBJ\rangle$ of the packets that make up the object. However, it is possible that S_{obj} might be so large that the size of $BV\langle OBJ\rangle\rangle$ might consume too much storage space within a resource constrained node. Under these circumstances, we adopt the approach taken by [5]. The data object *OBJ* is fragmented into P pages where each page has a fixed number of packets of size S_{pkt}, and each node only needs to maintain a bit vector for the current page being transferred.

IV. The McTorrent Protocol

In this section, we describe the McTorrent protocol for reliable bulk data distribution in sensor networks. This protocol is designed to use multiple communication channels while reliably distributing a large data object to all the nodes in the network.

A. Overview

The design of McTorrent shares many ideas with Deluge and MNP. The object to be disseminated is divided into pages, and each page in turn consists of a fixed number of packets. The object is disseminated in a pipelined hop-by-hop fashion, whereby a node that has received an entire page becomes a source node for propagating the page to its neighbors. The pages of an object are transferred in sequential order. Nodes send periodic advertisements that include the version number of the object and the number of pages it has available for transfer. A node that receives an advertisement for a page that it does not possess responds with a request that includes a bit vector indicating which packets in the page it needs. Nodes receiving requests then broadcast the requested data.

The key difference between McTorrent and its predecessors is in the use of multiple communication channels. The motivation for using multiple channels is twofold. First, by enabling different nodes to use different channels for transmitting data packets, we can reduce the number of collisions and reduce the latency of data propagation. Second, to ensure that transfers of different pages do not interfere with each other, Deluge takes special care to ensure that nodes sending simultaneously are at least three hops apart (see Figure 1). However, by using different data channels, senders only two hops apart can send packets at the same time in McTorrent as shown in Figure 2, thus enabling more effective spatial multiplexing.

Fig. 1. Pipelining in Deluge

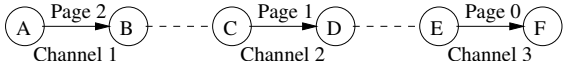

Fig. 2. Pipelining in McTorrent

Two issues need to be addressed in a data dissemination protocol that uses multiple channels. First, nodes need to advertise and request data from each other. Protocols such as Deluge exploit passive listening, whereby nodes suppress their own advertisements or requests on overhearing transmissions from neighboring nodes; this is also desirable for McTorrent. Second, the availability of multiple channels implies that nodes exchanging data need to agree on the channel used for a particular exchange, as well as the timing of the exchange. We address these issues by extending the basic advertisement-request-data approach. To benefit from passive listening, all nodes send their advertisements and requests on a common channel. However, data packets are sent on different channels.

B. Protocol Description

In a multichannel sensor network, each node is able to switch to any of a fixed number of available channels, and send and receive packets on that channel. For convenience, we use Γ to denote the number of available channels including the shared common channel γ_0. McTorrent uses four types of messages: advertisement (ADV), request (REQ), channel (CHANNEL), and data (DATA). The first three types of messages (control messages) are always sent on γ_0, while DATA messages are sent on a prespecified channel.

McTorrent adopts a four-phase ADV-REQ-CHANNEL-DATA exchange to propagate bulk data. In each four-phase exchange, a node who has new pages to send (called a sender) sends an ADV packet, and CHANNEL and DATA packets upon request. Nodes who need the pages (called receivers) send a REQ packet on reception of the ADV packet. The ADV packet contains the source address of the advertiser, summary information of the objects, and the channel (γ_j) on which it will send the DATA packets. γ_j is selected such that collisions of data packets are minimized. The channel selection algorithm is presented in IV-C. The REQ packet contains the destination address, the requested page number, and a bit vector of requested packets of that page. The channel number received from the ADV packet is echoed in the REQ packet.

The CHANNEL packet contains the same source address and channel number as the corresponding ADV packet, and additionally the page number and the bit vector of packets of the page the sender is going to send.

The McTorrent protocol can be described in terms of a set of local rules executed by nodes. Nodes can be in one of three states with respect to the operation of the protocol: idle, transmit, or receive state. Idle nodes, i.e., nodes that are neither senders nor receivers, periodically broadcast advertisements. We adopt the rules used by a node in Deluge [5] for dynamically adjusting the rate at which a node sends advertisements. A node transitions to the transmit state if it receives a request for one of the pages it possesses. It returns to the idle state after having broadcast all the requested packets.

Rules for a sender S:

1) On reception of the first request, S broadcasts a CHANNEL message containing the number of the requested page, the bit vector of requested packets and the channel number γ_j. After sending the message, S tunes its radio to γ_j.
2) S broadcasts all of the requested data packets on γ_j and tunes back to γ_0.

A node in the idle state transitions to the receive state if (i) it receives an advertisement for a page that it needs, or (ii) it receives a CHANNEL message with a page number and bit vector indicating that the sender will be transmitting packets that the node needs. The second scenario can occur if the node has missed the advertisement message.

A node in the receive state transitions back to the idle state if (i) if it has received the entire page being transmitted, or (ii) it has not received any packets for a time interval which is equal to the transmission time of k packets. The second scenario can occur if packets are repeatedly lost because of poor connectivity. (In our experiments, we set $k = 8$.)

Rules for a receiver R:

1) On receiving an advertisement from S for a page that it needs, R waits for a random time interval, T, which is uniformly distributed in the range (0, $T_{adv/2}$) where $T_{adv/2}$ is the minimum interval between advertisements. During time interval T, if it does not overhear a request sent to S, it sends a request containing the page number, a bit vector of desired packets and γ_j. After sending the request, R tunes to γ_j and starts a timer T1. The duration of T1 is set to the total transmission time of k data packets.
2) During time interval T, if R overhears a request sent to S, it tunes to channel γ_j and starts the timer T1.
3) If a node receives a CHANNEL message from S, and its packets-to-receive bit vector intersects with the packets-to-send bit vector in the message, it becomes a receiver of S if it is not one already. The node then tunes to γ_j and starts the timer T1.
4) If a receiver R receives a data packet from S, but the data is for a page that it does not need, R tunes to γ_0 and returns to the idle state.
5) If R receives from S a data packet of the page it needs, it restarts T1.
6) When timer T1 expires, or R has received the entire page, it tunes to γ_0 and returns to the idle state.

C. Channel Monitoring and Selection

In McTorrent, before sending an advertisement, the sender must select a channel for the imminent data transmission. Ideally, the sender should select a channel that does not conflict with the channels being used by senders one or two hops away. We use a contention-based approach in which a node monitors the ADV and CHANNEL messages broadcast by nodes in its neighborhood to keep track of the channels in use. It then attempts to select an unused channel for its own transmissions, or if there is no such channel, to use the channel that was used least recently by another node. We use a sliding window scheme to monitor the channel use in a locality and to select the least recently used channel. Although this approach does not guarantee that there will be no channel conflicts between nodes that are transmitting data, it is simple and works well if the number of senders in a network neighborhood is small relative to the number of receivers, which is typically the case for a data dissemination protocol.

More specifically, the channel monitoring and selection scheme is described below:

1) Every node maintains a counter for each possible channel [1..Γ-1] that is initialized to 0.
2) When a node X overhears a request containing channel γ_j, if the request is not destined to X, nor to X's source if X is in the receive state, the counter of γ_j is set to l, where $l \geq 2$.
3) When X overhears a channel message from S containing channel γ_j, if X is not a receiver of S, the counter of γ_j is set to l, where $l \geq 2$.
4) The counters are periodically decremented by 1 if they are greater than 0.
5) When choosing a channel, X randomly selects one from among the channels whose counters are 0.

The interval to refresh the counters is set to the time length of transmitting a whole page. Thus, a channel marked as being in use by another node will be cleared in at most l refresh intervals (in our experiments, we used $l = 2$).

D. McTorrent Evaluation

We used the TOSSIM simulator [19] to evaluate the performance of McTorrent. For our experiments, we modified the radio model in TOSSIM to support multichannel functionality. We used two metrics to evaluate the performance of McTorrent. The first metric is completion time, which is the time taken to reliably distribute the data object to all the nodes in the network. The second metric is the total number of packets transmitted while propagating the object. This metric reflects the energy consumption for data propagation.

In our simulation experiments, we evaluated these metrics for propagating a data object divided into five pages, where

each page consists of 24 packets and each data packet has a 23 byte data payload. We considered several networks of different sizes where nodes are deployed in a grid topology. We varied the spacing between nodes to examine the effect of node density on our results. We assume that initially only the base station, which is the node at the lower left corner of the grid, has the data object. The network scenarios and parameters considered in our simulations are identical to those used for some of the simulation experiments reported in [5].

To evaluate the impact of multiple communication channels, we considered various configurations of McTorrent with different numbers of available channels. Specifically, we considered configurations with the number of available channels Γ equal to 1, 4, and 16 (denoted as McTorrent-Γ). We also considered a channel configuration (denoted by $\Gamma = x$) in which each node was statically assigned a unique data channel. This configuration represents an ideal case in which there is no contention between nodes for channels. We also simulated the Deluge protocol for comparison purposes, and in the results below we report on the performance of both McTorrent and Deluge. Each simulation experiment was run multiple times with different random number seeds. 95% confidence intervals were obtained for the average values reported and are shown in the figures, except in the case of 25x25 and 30x30 grid networks, where the long simulation time did not give us a sufficient number of runs to obtain confidence intervals.

In our first experiment, we report on the time taken to disseminate five pages over a 10x10 grid network, where nodes are spaced 10 feet apart. Figure 3 shows the time for each node to receive all of the five pages, averaged over nine runs with different seeds. We see that the completion time at each node for McTorrent (with four or more channels) is lower than that for Deluge and McTorrent with a single channel. The completion time for Deluge and McTorrent-1 are comparable, reflecting the fact that they are essentially the same protocol. McTorrent-4 and McTorrent-16 have a larger completion time than McTorrent-x, showing the impact of contention between nodes for communication channels.

Next, Figure 4 shows the average number of packet transmissions over the nine runs. In comparison to Deluge, although McTorrent introduces about 1000 CHANNEL packets, for configurations with more than 4 channels, the number of advertisements, requests and data packets is reduced by more than 1000 packets each.

The savings in transmissions become more significant when the network size increases, as shown in Figure 5. In this figure, we plot the number of control and data packets for Deluge and McTorrent-16 for different network sizes. The control packets in the figure include all of the advertisements, requests, and channel packets. For a network of 400 nodes, the total number of data packets transmitted is 60% lower for McTorrent-16 than Deluge, and this trend continues for larger networks.

Figures 6 and 7 show the completion time and packet transmissions of Deluge and McTorrent for 20x20 networks with different densities (different grid spacings). As expected, the advantages of multichannel functionality decrease when

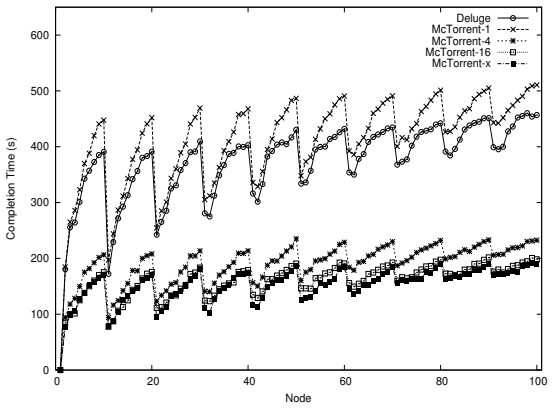

Fig. 3. Completion time for disseminating 5 pages to each node in a 10x10 grid network with inter-node spacing of 10 ft

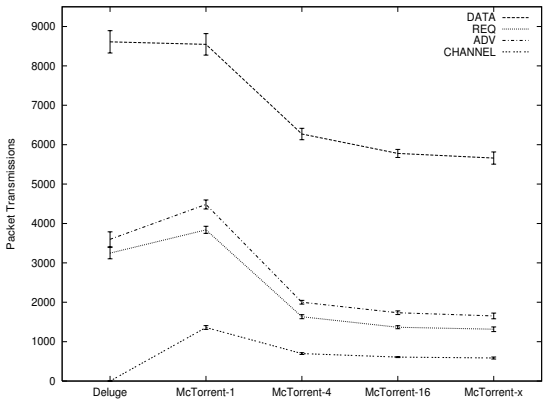

Fig. 4. Total number of packet transmissions for disseminating 5 pages in a 10x10 grid network with inter-node spacing of 10 ft

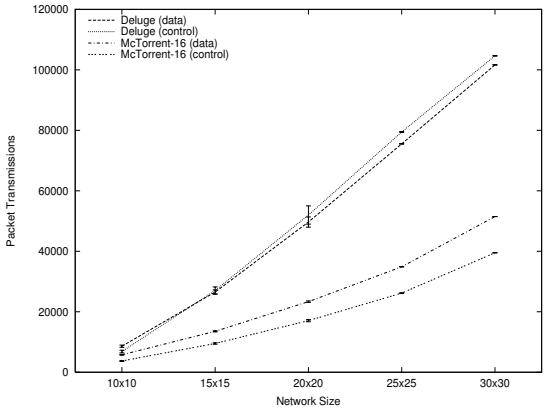

Fig. 5. Total number of packet transmissions for different network sizes (5 pages)

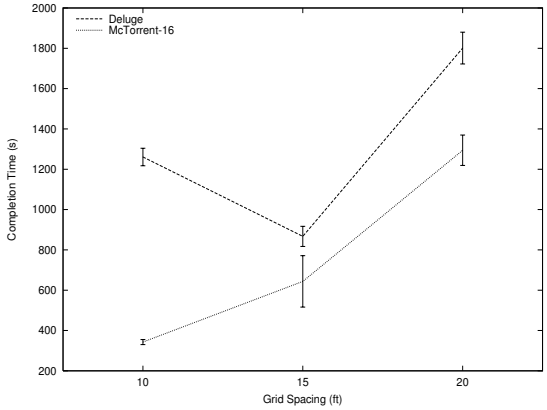

Fig. 6. Completion times for different network densities (5 pages)

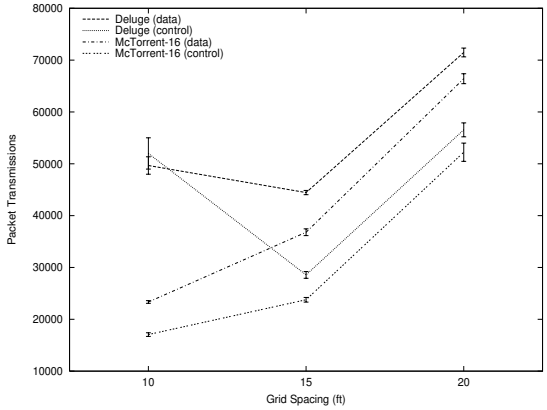

Fig. 7. Total number of packet transmissions for different network densities (5 pages)

networks get sparser, since the number of collisions is reduced. For Deluge, the completion time is lower when the nodes are spaced 15 feet apart than it is when the nodes are spaced 10 feet or 20 feet apart. This is because for the number of packet collisions is reduced as the spacing is increased from 10 to 15 feet. As the spacing is increased further to 20 feet, the number of packet retransmissions needed increases since the packet loss rate increases with distance between nodes. For McTorrent-16, the completion time increases as the spacing increases, reflecting the fact that packet collisions play a smaller role in its performance.

Overall, our simulation results show that the McTorrent approach leads to a significant reduction in the time taken to reliably propagate a large data object through a multihop network, and reduces the number of transmitted packets. There are two reasons for this improvement in performance. First, the number of collisions is reduced. Second, the multichannel capability leads to more effective spatial multiplexing and pipelining as shown in Figures 1 and 2.

V. McSynch

The McSynch protocol is designed to provide efficient local data synchronization and dissemination. In contrast to McTorrent, which provides reliable end-to-end data dissemination within the network, McSynch is used for cases where nodes within a few hops of each other have inconsistent or out-of-date data. This situation may arise under several circumstances, such as when sleep scheduling prevents nodes from receiving timely updates, when new nodes are added to an area, or when an initial end-to-end data dissemination phase failed to synchronize the entire network. During these situations, sensor nodes should be able to coordinate in a localized fashion to achieve data synchronization. We assume that such a synchronization decision can be made either by an external base station, or by the nodes themselves.

The advantage of the local synchronization approach used in McSynch is twofold. First, McSynch local recovery operations are performed without adding any significant communication overhead to the rest of the network. Second, McSynch uses a time-synchronized scheduling mechanism for time-bounded and collision-free data dissemination, thus minimizing the amount of time required for local synchronization.

A. Cluster formation

The McSynch protocol assumes that the network can be organized into a series of clusters, each with a designated clusterhead. After McSynch executes, all cluster nodes have a consistent and synchronized view of a common data object or value. During cluster formation the clusterhead also performs a time synchronization step. Notice that this step only requires local, cluster-wide time synchronization, and need not be maintained once the protocol executes. Any one of a number of proposed time synchronization protocols can be used, including [20].

McSynch forms two-hop clusters, with the clusterhead in communication range of all nodes. In order to prevent transmission interference from adjacent clusters we assume that clusterhead-to-clusterhead coordination is possible. Efficient clusterhead selection and clusterhead to clusterhead communication can be performed using one of several proposed approaches, such as [21]. The majority of packet transmissions do not involve the clusterhead which therefore does not become a bottleneck.

The outcome of the cluster formation step is that every node obtains from the clusterhead a unique local number between 1 and N. Further, during cluster formation, we require that all nodes obtain bit vectors from all the other nodes in the cluster. Since the bit vectors need only propagate within the cluster, this step does not require excessive overhead or time. We also assume that each node learns about its one-hop neighbors during the process of cluster formation.

B. McSynch Synchronization

After the cluster is formed, each cluster node has the bit vector $BV\langle OBJ\rangle$ from every other node. At this point, McSynch shifts operation and performs collision-free spatial multiplexing by assigning specific packets to particular time slots and channels. The scheduling algorithm assures that each

data packet needed by a node in the cluster has a *winning* node assigned to transmit that packet.

McSynch uses a round-based approach. At the end of each round, all packets required by other nodes have been transmitted once. This does not ensure that all nodes will have received all the packets that they need, since each node can only send or receive one packet at a time. In each round the packets associated with the data object are assigned to unique channels and time slots. Assume that the data object has Z packets associated with it, indexed by $0 \leq i < Z$. This also means that the bit vector BV has a length of Z bits. Further, assume that there are Γ channels. Write the packet associated with the ith portion of the data object as b_i. Packet b_i will be transmitted during the jth timeslot of that round, where $j = \lfloor \frac{i}{\Gamma} \rfloor$. Further, packet b_i is transmitted on channel γ_k, where $k = i \bmod \Gamma$. Each node determines, without any more data exchange, which node *win(b_i)* is to transmit b_i. The packet will be transmitted only if it is needed by at least one of *win(b_i)*'s one-hop neighbors.

Figure 8 shows the algorithm. Upon receiving an initiation code from the clusterhead, each node independently executes the algorithm. As shown in line 2, each round lasts for T time slots, where $T = \lceil \frac{Z}{\Gamma} \rceil$. Note that the final round may have some channels with no assigned packets. `NEED(`b_i`)`, `HAVE(`b_i`)`, and `NODES` are sets containing nodes. In each time slot, a total of Γ simultaneous transmissions can take place, one packet per channel. This is the meaning of the loop starting at line 4. In line 13 the function *HASH(BV_j,j, k)* is a hash function that takes as input the bit vector for node j (treated as a string), the identifier j and the channel number k.

This fully distributed algorithm produces collision-free scheduling, because only one node per channel can be designated to the variable *win(b_i)* at line 12. The node assigned to *win(b_i)* is the only node allowed to transmit on channel $i \bmod \Gamma$ for that timeslot. At the end of each round it may be the case that a node still does not have some required packets. This case can arise because none of its one-hop neighbors had the packet, or because it was either transmitting or receiving another packet on a different channel during the time slot that the packet was assigned, or simply because of packet loss. To account for this, the McSynch protocol reruns the bit vector exchange portion of the cluster formation step followed by another iteration, if required.

Algorithm McSynch Scheduling

1. Repeat until object synchronized
2. For t = 0 to $(T-1)$
3. `NODES` $\Leftarrow$ all nodes in the cluster
4. For $i = \left(\lceil \frac{t}{\Gamma} \rceil \times \Gamma\right)$ *to* $\left(\left(\lceil \frac{t}{\Gamma} \rceil \times \Gamma\right) + (\Gamma - 1)\right)$
5. Assign packet b_i to channel γ_k, where $k = i \bmod \Gamma$
6. `NEED(`b_i`)`, `HAVE(`b_i`)` $\Leftarrow$ `null`
7. `NEED(`b_i`)` $\Leftarrow$ all nodes $\in$ `NODES` that need b_i
8. If `NEED(`b_i`)` == `null` then skip to next i
9. `HAVE(`b_i`)` $\Leftarrow$ all nodes $\in$ `NODES` that have b_i and are 1-hop neighbors to at least one node in `NEED(`b_i`)`
10. If `HAVE(`b_i`)` == `null` then skip to next i
11. For all nodes j in `HAVE(`b_i`)`
12. Assign *prio(j)* to node j where *prio(j) = HASH(*`BV`$_j$`,` j`,` k*)*
13. Assign the node j with the highest *prio(j)* to *win(b_i)*
14. Node j transmits b_i on channel γ_k
15. `NODES` = `NODES` – (*win(b_i)* $\bigcup$ `NEED(`b_i`)`)
16. END for i
17. END for t

Fig. 8. The McSynch Protocol, simultaneously executed at each node

C. McSynch Evaluation

We performed an extensive experimental evaluation of McSynch. The purpose was to evaluate the speedup due to our spatial multiplexing approach. Our method was to focus on the number of slots required for complete synchronization. Since McSynch uses time-synchronized scheduling, packet collisions do not occur. Therefore, in order to focus on the time effectiveness of the algorithm, we used a custom simulator that assumed a perfect physical layer.

The experiments examined a number of different cluster scenarios. All clusters had a diameter of two hops. We ran experiments for 10, 15, 20, 25, and 30 nodes. The experiments evaluated clusters with sparse and dense connectivity. A cluster is considered *sparse* if each node has, on average, 35% of the nodes as one-hop neighbors. A cluster is considered *dense* if each node has, on average, 75% of the nodes as one-hop neighbors.

We tested objects of various lengths, but for space reasons only report on the results for a 96 packet object. However, our reported results are consistent across object size. We considered two different patterns of data availability. The first assumed that all nodes had a uniform likelihood of having a particular packet. The second type assumed that some nodes had a much higher likelihood of having packets than other nodes. The uniform pattern can be used to represent clusters where nodes have relatively similar sleep cycles that might cause them to miss packets, or where previous transmission impairments were uniform. The second pattern type, which we call uneven sleep, can represent clusters that have unbalanced sleep patterns (perhaps for energy conservation) or the case where some nodes are added to the network after other nodes. The uneven sleep scenarios had 25% of the nodes having on average 90% of the packets, while the remainder of the nodes had on average 20% of the packets.

Because of the way McSynch uses time synchronization, it can not be directly compared to other protocols such as McTorrent, Deluge or SPIN-BL. However, as a point of reference, we propose and evaluate a single channel time-slotted request-reply protocol, abbreviated TS-RR, which is similar to

Trickle [16]. Trickle is designed to use suppression timers to reduce the number of advertisements, requests, and replies. For purposes of our evaluation, we assume that all advertisements, requests and replies can occur with *no* collisions. Likewise, we assume that the time to perform McSynch cluster-wide bit vector exchange is constant. We therefore do not factor in the cost of advertising or re-advertising the bit vectors into our evaluation. This puts TS-RR at an advantage compared to McSynch, since McSynch does not require per packet requests.

Each scenario was run 100 times using a different random number generator seed. The results shown are the average time to complete with 95% confidence intervals. The requests are spread reliably throughout the cluster. The first two results show the latency to completion for the case of a 96 packet object size with a uniform sleep pattern for sparse (Figure 9) and dense networks (Figure 10). Increasing the number of nodes increases the time to complete, since more transmissions are required. We also see that increasing the number of channels available to McSynch decreases the total time to complete. The time to complete for 16 channels stays roughly the same for all network densities and number of nodes. This shows that with the appropriate number of channels McSynch significantly reduces the time to perform local synchronization.

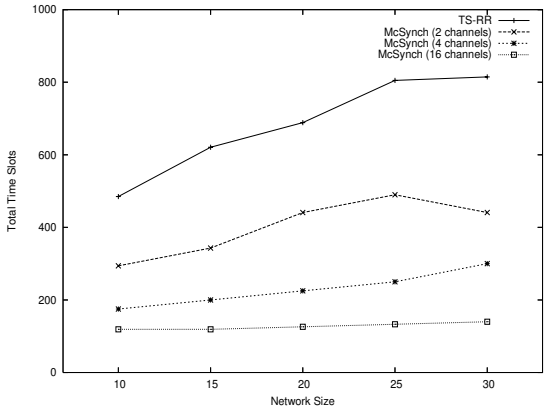

Fig. 9. Time slots required for object synchronization for different network sizes, with sparse network connectivity and uniform node sleep patterns

Figures 11 and 12 show the time to complete for the case of a sparse network and a dense network with nodes having uneven sleep patterns. Compared to the uniform sleep patterns in the first set of results, the uneven sleeping patterns increase the time to complete for all scenarios. Despite this fact, note that the 16 channel case still produces a roughly constant time to completion, independent of the network density and number of nodes. These experiments allow us to conclude that the multichannel McSynch approach significantly outperforms single channel request-reply approaches. Further, by using a sufficient number of channels the time to complete is constant for two-hop clusters up to at least 30 nodes.

VI. Multichannel implementation study

We implemented a multichannel test environment within a Crossbow MICA2 MPR400 network. The testbed software

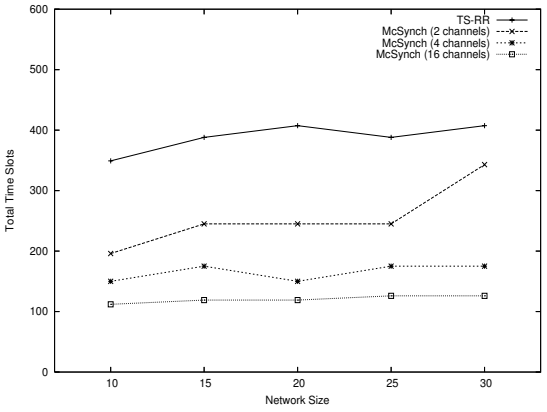

Fig. 10. Time slots required for object synchronization for different network sizes, with dense network connectivity and uniform node sleep patterns

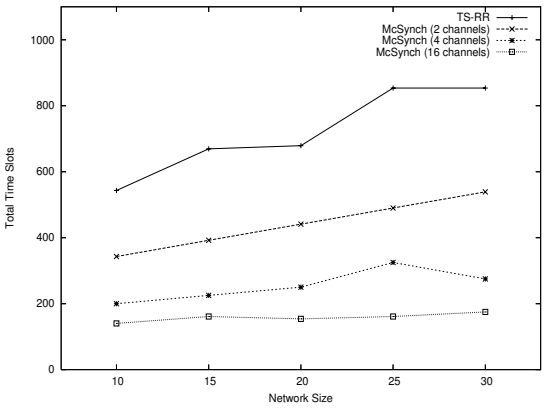

Fig. 11. Time slots required for object synchronization for different network sizes, with sparse network connectivity and uneven node sleep patterns

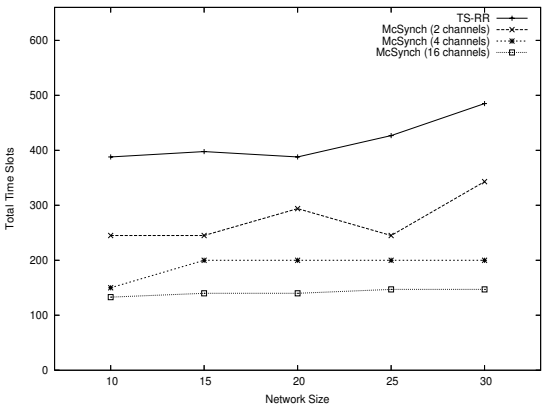

Fig. 12. Time slots required for object synchronization for different network sizes, with dense network connectivity and uneven node sleep patterns

uses a node-resident API that can adjust operational aspects of a CC1000 radio. In addition to basic operations like altering transmit power and reporting signal strength, the API implements complex operations by extending CC1000 code. These operations include toggling sender backoff, reporting partial packet receptions, and extracting RF noise levels. In addition, the API triggers transceiver power cycling to allow for realtime radio channel selection.

The nature of wireless communication introduces radio irregularities that affect packet reception and protocol performance [14]. These irregularities are caused by factors including antennae orientation, multipath effects, location, and power disparity. Unfortunately, the introduction of multichannel communication adds adjacent channel interference as yet another cause of radio irregularity. In order to measure the influence of channel interference on packet reception, we used the test environment we developed to measure interference at different channel spacings. Our results enable us to determine the minimum channel spacing that prevents interference, and thereby maximize the number of available channels for radios with bounded frequency regions like the CC1000.

A line experiment consisting of two senders and a receiver is configured as follows. A sender is instructed to send p packets on channel frequency F. The other sender is instructed to synchronously send p packets on a channel frequency F' = F - 1MHz. A receiver is placed equidistant from the two senders and instructed to receive packets from one of them. The experiment is repeated n times, where frequency F' is incremented by $\frac{2}{n}$ MHz each time. The set of experiments is then repeated with the receiver instructed to listen to the other sender.

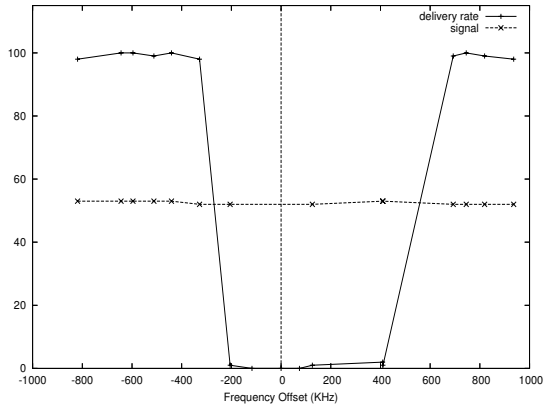

Fig. 13. Adjacent channel interference

In the figures that follow, reception rate and signal strength do not share the same scale, but are shown in the same graph to illustrate the correlation between them. The graph in Figure 13 depicts the packet reception rate and received signal strength for an experiment with a 909.98MHz center frequency. A data point represents the packet reception rate for a 200 packet set. As expected, interference occurs for small channel spacings. The solid line indicates that no packets were received when

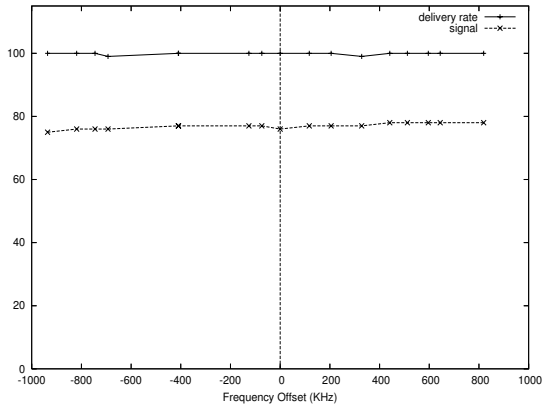

Fig. 14. Minimal interference

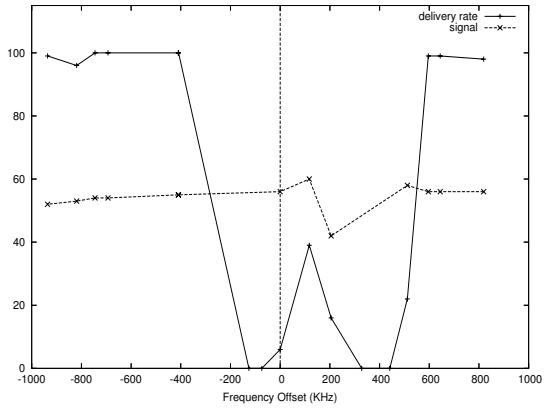

Fig. 15. Interference anomaly after moving weaker sender

the two data channels were up to 400KHz of one another, as portrayed by the trough from -200KHz to 400KHz. Further experiments showed the minimum channel separation that avoids collisions to be 500KHz. Given the 26MHz frequency region of the MPR400, this limits the number of contention-free radio channels to around 50. The graph also shows a dashed line representing normalized values for the received signal strength recorded by the radio. A data point is the average of the normalized signal strength for any of the 200 packets received. If no packets are received, however, the signal strength is interpolated. In Figure 13, the known signal strength is relatively constant.

The graph in Figure 14 shows the identical experiment, but with the receiver counting transmissions from the other sender. Although the first experiment confirmed that adjacent channel interference occurs for small separations, this second graph shows near perfect reception for any channel spacing. The reason lies in the signal strength disparity between the two senders. The second sender has a stronger signal than the first. As a result, its data packets are unaffected by the interference from its weaker counterpart.

The graphs in Figures 15 and 16 plot packet reception when the received signal strength from the two senders is more

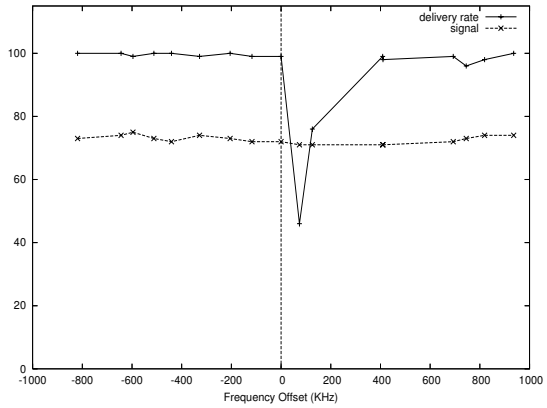

Fig. 16. Interference anomaly after further moving weaker sender

comparable. As compared to the experiments shown in Figure 13, the signal strength difference is reduced by moving the weaker node closer to the receiver. These graphs are similar to the first two, except for an anomaly which occurs when the channel separation is about 100KHz. The graphs show a moderate increase in reception from the weaker node, and an equivalent drop in reception from the stronger node. The cause of this anomaly is not obvious, but the unusual signal strengths at 100KHz and 200KHz in Figure 15 are indicative of irregular radio effects. These effects are a combined result of the reflection, refraction, and diffraction a signal experiences as it propagates from a transceiver. The effects can be constructive or destructive depending on the specific locations of the sender and receiver in a given environment. If constructive effects are the cause of the anomaly, the received signal strength should be greater than its normal level when the anomaly occurs. Indeed, in Figure 16 we see that this effect becomes even more pronounced as the weaker sender is moved even close to the receiver.

VII. Conclusions

In this paper, we described *McTorrent*, a protocol for multihop data dissemination, and *McSynch*, a cluster-level protocol for localized data synchronization and coordination. McTorrent is similar to Deluge in its approach to data management. The major difference, and the reason why McTorrent outperforms Deluge, is the way it uses multiple communication channels. Our simulation results show that McTorrent achieves end-to-end data dissemination in less time than the single channel protocols. McSynch operates on a cluster-wide basis and uses a distributed scheduling approach for channel access. Once cluster information is exchanged between the cluster nodes, all protocol actions are entirely distributed. Our simulation results showed that by using an appropriate number of channels McSynch can substantially reduce the time required cluster-wide synchronization.

We also report on the implementation and analysis of a multichannel prototype within a sensor node testbed. Our results indicate that a primary factor in determining performance in a multichannel setting is channel separation, in addition to the signal strength. We also observed several physical layer anomalies due to constructive multipath effects.

References

[1] K. Pahlavan and P. Krishnamurthy, *Principles of Wireless Networks: A Unified Approach*. Prentice Hall, 2002.

[2] "MICA2 radio stack for TinyOS." [Online]. Available: http://www.tinyos.net/tinyos-1.x/doc/mica2radio/CC1000.html

[3] P. Levis *et al.*, "The emergence of networking abstractions and techniques in TinyOS," in *Proceedings of the 1st USENIX/ACM Symposium on Networked Systems Design and Implementation (NSDI)*, 2004.

[4] T. Stathopoulos *et al.*, "A remote code update mechanism for wireless sensor networks," CENS, Tech. Rep., 2003.

[5] J. Hui and D. Culler, "The dynamic behavior of a data dissemination protocol for network programming at scale," in *Proceedings of the 2nd International Conference on Embedded Networked Sensor Systems (SenSys)*, 2004.

[6] S. Kulkarni and L. Wang, "MNP: Multihop network reprogramming service for sensor networks," in *Proceedings of 25th International Conference on Distributed Computing Systems (ICDCS)*, 2005.

[7] A. Nasipuri and S. Das, "Multichannel CSMA with signal power-based channel selection for multihop wireless networks," in *Proceedings of the IEEE Fall Vehicular Technology Conference (VTC)*, 2000.

[8] S. Wu *et al.*, "A new multi-channel MAC protocol with on-demand channel assignment for multi-hop mobile ad hoc networks," in *Proceedings of the 2000 International Symposium on Parallel Architectures, Algorithms and Networks (ISPAN)*, 2000.

[9] N. Jain *et al.*, "Multichannel CSMA MAC protocol with receiver-based channel selection for multihop wireless networks," in *Proceedings of the 9th International Conference on Computer Communications and Networks (IC3N)*, 2001.

[10] Z. Tang and J. Garcia-Luna-Aceves, "Hop reservation multiple access (HRMA) for ad-hoc networks," in *Proceedings of 18th Annual Joint Conference of the IEEE Computer and Communications Societies (INFOCOM)*, 1999.

[11] J. Garcia-Luna-Aceves and A. Tzamaloukas, "Receiver-initiated collision avoidance in wireless networks," *Wireless Networks*, vol. 8, no. 2/3, pp. 249–263, 2002.

[12] J. So and N. Vaidya, "Multi-channel mac for ad hoc networks: Handling multi-channel hidden terminals using a single transceiver," in *Proceedings of the 5th ACM International Symposium on Mobile Ad Hoc Networking and Computing (MobiHoc)*, 2004.

[13] A. Adya *et al.*, "A multi-radio unification protocol for IEEE 802.11 wireless networks," in *Proceedings of the International Conference on Broadband Networks (Broadnets)*, 2004.

[14] G. Zhou *et al.*, "Impact of radio irregularity on wireless sensor networks," in *Proceedings of the 2nd International Conference on Mobile Systems, Applications, and Services (MobiSys)*, 2004.

[15] J. Kulik *et al.*, "Negotiation-based protocols for disseminating information in wireless sensor networks," *Wireless Networks*, vol. 8, no. 2/3, pp. 169–185, 2002.

[16] P. Levis *et al.*, "Trickle: A self-regulating algorithm for code propagation and maintenance in wireless sensor networks," in *Proceedings of the 1st USENIX/ACM Symposium on Networked Systems Design and Implementation (NSDI)*, 2004.

[17] N. Reijers and K. Langendoen, "Efficient code distribution in wireless sensor networks," in *Proceedings of the 2nd ACM International Conference on Wireless Sensor Networks and Applications (WSNA)*, 2003.

[18] F. Stann and J. Heidemann, "RMST: Reliable data transport in sensor networks," in *Proceedings of the 1st International Workshop on Sensor Network Protocols and Applications (SNPA)*, 2003.

[19] P. Levis *et al.*, "TOSSIM: Accurate and scalable simulation of entire TinyOS applications," in *Proceedings of the 1st International Conference on Embedded Networked Sensor Systems (SenSys)*, 2003.

[20] S. Ganeriwal *et al.*, "Timing-sync protocol for sensor networks," in *Proceedings of the 1st International Conference on Embedded Networked Sensor Systems (SenSys)*, 2003.

[21] J. Wu and H. Li, "On calculating connected dominating sets for efficient routing in ad hoc wireless networks," *Telecommunication Systems, Special issue on Moible Computing and Wireless Networks*, vol. 18, no. 1/3, pp. 13–36, 2001.