Multi-Channel Data Collection for Throughput Maximization in Wireless Sensor Networks

Ying Chen, Pedro Henrique Gomes and Bhaskar Krishnamachari
Viterbi School of Engineering, University of Southern California, USA
Email: {chen2,pdasilva,bkrishna}@usc.edu

***Abstract*—We present the design and implementation of Multi-Channel Collection (MCC) protocol , a high-rate multi-channel time-scheduled protocol for fair, real-time data collection in Wireless Sensor Networks (WSN). MCC incorporates sophisticated mechanisms for balanced routing tree formation, multiple frequency channel allocation and globally synchronized TDMA scheduling. Through systematic experiments with real WSN hardware (Tmote Sky), we identify the maximum possible throughput for many-to-one (convergecast) data collection as a function of key communication parameters such as packet size, use of acknowledgements, and network topology. Then, we demonstrate that the maximum achievable network throughput can in fact be attained in practice using a carefully designed mix of routing, frequency allocation and time scheduling. Compared to state of the art collection protocols for WSN, we show that MCC offers 33-155% improvement in throughput. We also show how to exploit the time-scheduled nature of this approach for reducing the number of required frequency channels. MCC presents an algorithmic approach for time-frequency scheduling and routing that could be adapted and used in conjunction with relevant emerging standards such as WirelessHART, ISA 100.11a and IEEE 802.15.4e TSCH.**

***Keywords*—Wireless sensor networks; Collection protocol; IEEE 802.15.4e TSCH; WirelessHART**

I. Introduction

Networks of wireless sensors, each capable of a combination of sensing, computation, radio communication, possibly even actuation, are envisioned to form a key component of the emerging "Internet of Things". Wireless Sensor Networks (WSN) have been developed and put into use for automated data collection in many different scenarios, like environment monitoring, traffic monitoring, building automation, etc.

In many real implementations [1], [2], it has been found that even if each sensor generates low rate data individually, due to the density of deployment, the many-to-one hop-by-hop traffic pattern and the lack of data compression/aggregation, the amount of traffic close to sink is still very high, leading to high loss rate and poor throughput. This is even true for WSNs with small size and light traffic as shown in [2]. For example, as mentioned in [3], experiments on a testbed using Tmote Sky devices showed that with 40-byte packets, the per-source rate in a 40-node WSN is only about 0.5pkts/sec; thus the network throughput is only $6.4kbps$, about 2.56% (!) of the nominal $250kbps$ data rate of the IEEE 802.15.4 radio on Tmotes. *These observations illustrate that sensor networks are fundamentally throughput limited.*

This work was supported in part by NSF via awards numbered 1049541 and 1201198

This work is therefore aimed at improving the throughput performance of wireless sensor networks. We ask and answer two questions based on experiments with state-of-the-art wireless sensor network hardware : 1) What is the maximum possible data collection rate for a given network? 2) Can this maximum rate be achieved in a real system?

To answer the first question, we first undertake a series of experiments on a representative widely-used WSN platform, the Tmote Sky. We study how throughput performance is affected by the packet size, node function (leaf / relay / sink), and the use of acknowledgements for reliability. Our experiments show that the maximum achievable throughput varies significantly with these parameters, and is well below the radio link rate provided by the PHY layer. To answer the second question, we design and implement the first-ever real protocol implementation of fair throughput-maximizing multi-channel convergecast, and demonstrate that it is able to achieve almost the highest-possible rate given the hardware limitations of a real sensor node.

Significant effort has been expended in recent years on developing and experimentally evaluating high throughput routing and rate-control protocols [3], [5]–[7]. Nearly all of these works have been single-channel, CSMA-based protocols. While these mechanisms are relatively easy to implement, their data rate is significantly limited by co-channel interference, particularly in dense deployments. Interference could be reduced by allocating different channels to different links, improving the efficiency of bandwidth usage. However, time synchronization is needed to efficiently coordinate the communication of nodes operating on different frequency channels. This motivates us to design and implement a time-scheduled multi-channel data collection protocol, that we refer to as *Multi-Channel Collection (MCC)*.

MCC is well-aligned with emerging new protocols and standards focused on industrial WSN, such as WirelessHART, ISA 100.11a and IEEE 802.15.4e (TSCH). We proposed a realistic and generic mechanism for implementing time-frequency scheduling, which we believe could be adapted to be used with these and other relevant protocols and standards.

We compare MCC with the *Collection Tree Protocol (CTP)* [8], a state of the art single-channel collection protocol for WSN, and with RCRT [7], a state of the art single-channel centralized rate allocation protocol. MCC provides significant improvements in throughput, ranging from 48% to 155% with respect to CTP and even more with respect to RCRT.

We summarize the key contributions of this work:

- Systematic experimental analysis and modeling of the impact of packet size, node function, and use of link-layer acknowledgements on the maximum achievable throughput in real WSN hardware.
- Empirical demonstration that it is possible to obtain close to the estimated maximum achievable network throughput in a real network.
- The finding that doing channel allocation using time scheduling information is effective in reducing the number of channels required for multi-channel protocols.

The rest of the paper is organized as follows. First, we discuss prior work in section II. In Section III, we model, analyze and measure the maximum achievable throughput of convergecast. In section IV, we present the design and implementation of MCC. In section V, we describe the test environment. In section VI, we evaluate the various components of MCC. Then, we deploy MCC and compare with state of the art single-channel collection schemes in section VII. Finally, we present future work in section VIII and summarize our contributions in section IX.

II. Related work

One of the objectives of our work is on characterizing the maximum achievable throughput in a WSN. This question was also posed by Österlind and Dunkels [10] in the context of a linear (single branch) network, motivated by bulk transfer protocols. The authors also consider the problem of the maximum achievable throughput and the role played by the Serial Peripheral Interface (SPI) bus transfer rate in linear (single branch) path. In our work, we investigate in a more comprehensive and fine-grained way this problem in a general network (multi-branch), including the consideration of the packet size, node roles, and ACKs. And we propose an analytical two-parameter model for maximum throughput estimation, which also accounts for non-packet-size-dependent factors such as software delays.

The number of papers written on collection and multi-channel protocols is too vast to enumerate comprehensively. Instead, we focus our review on key works, particularly those that have focused on experimental validation over testbeds.

The *de facto* state of the art data collection protocol for WSN today is the Collection Tree Protocol (CTP) [8], which is a single-channel protocol that offers the best and reliable delivery performance. We therefore use it as a baseline comparison for MCC. An alternate routing approach is the queue-aware dynamic routing BCP [5], which enhances throughput as well as robustness to external interference. Others have focused on avoiding congestion collapse and maintaining high-rate delivery by using a rate control protocol; examples of those are IFRC [4] and WRCP [3]. A centralized approach to rate control, RCRT [7], has yield even better rate performance. Flush [6] offers a robust, high-rate connection-oriented bulk transfer capability. All these protocols are single-channel protocols, and have been developed over CSMA. In this work, we compare the throughput obtained by our approach with that of CTP and RCRT, to understand how much improvement can be obtained with a multi-channel scheduling approach over single-channel CSMA-based schemes.

A comprehensive survey of multi-channel mechanisms for wireless networks can be found in [11]. MMSN [12] is the first multi-channel MAC protocol especially designed for WSNs with devices having half-duplex single transceiver. In MMSN there is a common broadcast channel, and nodes contend to access the channel on different unicast frequencies. TMCP [13] is a tree-based channel allocation mechanism, in which the tree is partitioned into separate sub-trees, each of which is allocated a separate channel. The authors in [14] show a cluster-based dynamic control-theoretic approach of multi-channel MAC design, which offers a distributed joint time and frequency scheduling mechanism.

Y-MAC [17] presents a multi-channel protocol that is based on lightweight channel hopping. Nodes on a link hop to a new channel when traffic bursts occur, following a predetermined sequence. The focus of Y-MAC is to improve energy efficiency by reducing contention, rather than high-rate performance, and it does not offer any guarantees in this regard.

PIP [16] is a joint TDMA-FDMA based bulk-transfer protocol. In its basic form, it allows the sink to establish a connection with a particular sensor node and download data from that node at the highest rate possible. By keeping the sink occupied half the time (it must remain idle whenever the node one hop from it is in receive mode) it can achieve at least 50% of the maximum throughput. P^3 [15] improves the performance of PIP using synchronized multi-path transmissions that completely fulfills the packet pipeline. In P^3's proposed architecture all intermediate nodes between source and sink are split into two sub-groups, each one working on a different channel, which permits that packets are transmitted and received continuously throughout the pipeline. Besides that, P^3 uses multi-transmitter constructive interference in order to increase the probability of packet reception.

A key distinction between the last two near-optimal multi-channel protocols and our proposal is that the MCC protocol is able to collect data fairly from all nodes in the network, rather than establishing connections only to one at a time. Thus, it is more suitable for real-time, fair data gathering application.

TSMP [23] is a centralized multi-channel mesh protocol designed specifically for high reliability. It is currently used as the basis of WirelessHART and ISA100.11a standards for industrial wireless communication, and was used as basis for the amendment IEEE 802.15.4e (TSCH). TSMP offers reliability of 99.9% and frequency hopping, which makes it efficient for noisy environments and critical applications. The focus of TSMP and other protocols designed for harsh environments is on the reliability of delivery, while MCC targets at scenarios requiring high data rate. TSMP offers a relatively lower maximum theoretical throughput of 76Kbps [23] and MCC demonstrates in practice throughput of about 100Kbps, an improvement of 33%. MCC presents algorithmic details of a generic framework for scheduling independent of network topology, which could be adapted and extended to work in accordance to WirelessHART standard. Ultimately, frequency hopping algorithm, which strengthens TSMP against interference, can be easily adapted to be used on MCC. The frequency channels to be used by each node at time slots can be considered as virtual channels, which are used by MCC during time-frequency scheduling. As long as the frequency

hop patterns are carefully designed to be orthogonal, MCC can work with frequency hopping without drastic changes.

III. Maximum Throughput Estimation

In practice the maximum achievable throughput may be much lower than the data rate of the underlying radio, mainly due to hardware limitations. In this section, we systematically study one widely used WSN hardware platform, the Tmote Sky, and understand its achievable throughput.

A. Basic Model

We show in Fig. 1 a high level view of main component that are involved during packet transmission/reception.

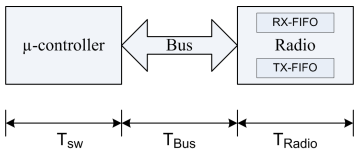

Fig. 1. A high level view of sensor node architecture

The node has a micro-controller, a radio with 2 separate buffers (TX and RX), and a bus to copy data between them. The total time for transferring a single packet, T_{Packet}, can be written as a linear combination of a constant and a term proportional to the packet size, as follows. As it will be clarified in III-B, depending on the role of the node being analyzed, T_{Packet} may include either TX or RX time, or both.

$$T_{Packet} = \alpha + \beta \cdot Packet\_Size \tag{1}$$

The parameter α pertains to per-packet software and radio latencies that are independent of the packet size. The rate term β depends on the CPU rate, the bus transfer rate, the radio rate, and the degree of pipelining achieved in successive transmissions. We see that α is very software-specific, while β can be estimated based on hardware specs for the Tmote Sky - it lies between 0.05 and 0.08 ms / byte. The corresponding throughput can then be calculated as follows.

$$Throughput = \frac{Packet\_Size}{T_{Packet}} \tag{2}$$

B. Estimating the Maximum Convergecast Throughput

In convergecast applications, nodes can be categorized, according to their roles in communication, into three types: sink (RX only), leaf (TX only) and relay (RX and TX). Depending on the role of a node, T_{Packet} and the corresponding throughput will be different. To measure them, we allocate time slots and synchronized transmissions in different topologies. To measure sink RX rate, we use a star-topology with the sink receiving packets from multiple one-hop nodes. To measure the leaf TX rate, we use a simple two-node link with the transmitter continuously sending packets. To measure the relay RX/TX rate, we consider a linear topology and measure the relay RX/TX rate of intermediate nodes as they alternate between sending and receiving packets in non-interfering channels. We vary the packet size from 10 bytes to 110 bytes and conduct the experiments with and without link-layer ACK. For each setting we vary the slot length to determine the maximum achievable rate and plot only those maximum rate values. The results obtained for the case without ACK is shown in Fig. 2. For the ACK-enabled case, the curves were very similar to the curves of Fig. 2, only changing the values of maximum data rate. The corresponding parameters of the best-fit parameters for α and β are shown in Table I, for both ACK-enable and ACK-disable cases. *We find that the measurements show an excellent fit with the throughput model described in equations (1) and (2).* The proposed model will be used for comparing the experimental results with theoretical maximum throughput.

We observe that the maximum achievable throughput in practice is at most about 100 kbps, well below the ideal 250 kbps link rate provided by the CC2420 radio.

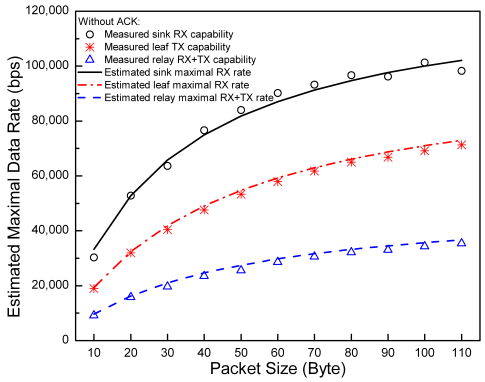

Fig. 2. Measured versus estimated throughput of Tmote Sky device for ACK-disabled case

TABLE I. Parameters for the curve-fit of throughput estimation model

	α (ms)	β (ms/byte)
Sink, no ACK	1.79	0.062
Relay, no ACK	3.50	0.079
Leaf, no ACK	3.35	0.079
Sink, ACK	3.95	0.078
Relay, ACK	5.81	0.085
Leaf, ACK	5.52	0.079

The obtained results yield two key observations that inform our design of a high-throughput collection protocol:

- The throughput performance is a concave function of the packet size.
- The throughput performance is affected significantly by the role of a node. We find that the maximum sink RX rate is slightly higher than the maximum leaf TX rate. This is due to the possibility of greater pipelining at the sink which is receiving data from multiple transmitters[1]. We also find that the relay RX/TX rate, which is already halved as it must both receive and transmit, is further reduced due to the switching overhead between frequency channels. Thus, in a linear

[1]At the transmitter, the application must wait to send the next packet from the microprocessor to the TX buffer on the radio until it is notified of the previous packet's transmission. From a receiver's perspective, the packet from one transmitter may overlap the time that the previous packet from a different transmitter is being copied from the RX buffer to the microprocessor.

topology, the throughput bottleneck would be the relay node's maximum throughput. However, in our case (tree topology), where the aggregated rate of multiples branches may exceed the maximum sink rate, the throughput is bounded by the maximum sink RX rate.

IV. MCC Design and Implementation

Based on the previous section, we find that for convergecast, conceptually, there are three possible sources of bottleneck: (1) interference, (2) relay node RX/TX rate and (3) sink RX rate. As argued in [9], we can mitigate (1) using multiple channels. We can address (2) by using suitably balanced routing topology. Therefore (3) becomes the fundamental limit on fair throughput, which MCC tries to optimize.

MCC includes network connectivity determination, routing tree formation, channel allocation, time synchronization, time scheduling and data collection, which are described below.

A. Network Connectivity Determination

This is the bootstrapping phase of MCC. The PRR values are first computed in a distributed manner by each mote. Initially, all nodes in the network start in the same channel and broadcast 100 messages. Whenever a node receives a message, it logs the sender's ID, and uses these messages to compute the PRR for each other node it hears from. The messages are randomly broadcasted over a period of 1 minute. The obtained PRR values represent the link quality due to channel quality and external interferences.

The server uses the PRR values obtained along with a threshold parameter Θ to compute an undirected connectivity graph $G(V, E)$ and an undirected interference graph $I(V, E')$ of the network for use in routing and scheduling. An edge (i, j) is placed in G if and only if $PRR(i, j) > \Theta$ and $PRR(j, i) > \Theta$. An edge (i, j) is placed in I if either $PRR(i, j)$ or $PRR(j, i)$ are non-zero, in other words, all links with non-zero PRR are considered conservatively as potential interferers. Considering the undirected version of both graphs eases the implementation of the routing and scheduling algorithms.

It is clear that network connectivity is a very dynamic parameter of our system. In this work we consider that nodes are fixed and that the channel quality of all links does not change significantly, which means that graphs $G(V, E)$ and $I(V, E')$ do not change. It may be not true in real scenarios, where nodes are mobile or environment changes; in such scenarios MCC can be extended to overcome the network dynamics (as described in section VIII).

B. Balanced Routing Tree Formation using CMS

We assume that interference can be eliminated by allocating multiple channels in a TDMA network. It has been shown in [9] that the available schedule length is lower-bounded by $max(2n_k - 1, N)$, where n_k is the maximum number of nodes on any sub-tree and N is the number of nodes in the network. If it is possible to have $2n_k - 1 < N$ in the tree construction, we can achieve N as the lower bound for time scheduling. But for an arbitrary graph G, can we construct a tree T on G such that $n_k < (N + 1)/2$? This is a special case of the "Capacitated Minimal Spanning Tree Problem" that is known to be NP-complete [18]. For MCC, we use the Capacitated Minimal Spanning (*CMS*) Tree heuristic presented in [9] to build a balanced routing tree. Despite the general worst case complexity result for this problem, we find that in practice this heuristic can consistently build a tree that satisfies the condition $n_k < (N + 1)/2$. The basic idea of the CMS tree construction algorithm is to minimize the size of each branch by considering the possible growth that a node might bring to it.

C. Channel Allocation

We adopt a receiver-based channel allocation approach, whereby nodes allocated a channel use only that channel for receiving packets but may use another channel to transmit to their parent. The problem of channel allocation is essentially the NP-hard problem of graph coloring. We use the greedy degree-ordering heuristic known as Welsh-Powell [21] algorithm, which has been proven to perform well in practice. The input of the channel allocation is a conflict graph H and number of available channels. A conflict graph H means that if the edge (i, j) exists in H, then nodes i and j must be allocated different reception channels. We defer to section IV-E a description of different approaches to generate graph H.

D. Time Scheduling

We solve the time scheduling problem taking into account only the tree topology under the assumption that all interference as defined by the conflict graph H can be completely eliminated using channel allocation. Our time scheduling algorithm is modified from the algorithm indicated in [20], which is proved to derive minimum data collection time and optimal strategies on tree networks. Instead of solving the collection problem on a tree topology, the algorithm initially looks at a simplified converse problem, i.e., the *distribution problem* on a line topology. To distribute all data packets in the minimum time, the sink sends the first packet destined for the furthest node, then for the second furthest one and so on, as fast as satisfying the half-duplex feature of the radio. The key difference between our modified scheduling and that described in [20] is that while the original algorithm assumes same slot length for TX and RX, for MCC we double the length of RX slots on relay nodes in order to introduce a guard-time while keeping a tight TX slot so the sink could continuously receive packets. For our modified algorithm, we derive the equation (3) to find node i's last busy TS, where i represents the distance to the sink in a line and v_i represents the number of data packets stored at node i at the end of the observation period.

$$T_i = \begin{cases} 2 * (i-1) + 3\sum_{j \geq i+1} \nu_j, & \text{if } \nu_i = 0, \\ 2 * \nu_i + 3\sum_{j \geq 2} \nu_j, & \text{if } i = 1 \text{ and } \nu_1 \geq 1, \\ 2 * i - 3 + 3\sum_{j \geq i} \nu_j, & \text{if } i \geq 2 \text{ and } \nu_i \geq 1. \end{cases} \quad (3)$$

The building block in our algorithm is a slot, which can be of 3 types: TX, RX and ID (idle). The TX and ID slots have duration of one slot-length and the RX mode has duration of 2*slot-length. The sink node is scheduled to remain in RX mode at all times. Nodes follow the same schedule pattern in each period (corresponding to one round of data

collection from all nodes). It is important to note, though, that different network requirements can be accommodated on MCC architecture; a different number of slots might be allocated for a set of nodes that temporarily need more throughput. Also, the time slot allocation on MCC aims at guaranteeing the maximum fairness, but it can be adapted for networks with heterogeneous applications.

Applying equation (3), we can find node i's transmission time, based on the symmetric operations of the distribution problem on a line topology and collection problem on a tree topology. The algorithm can be further generalized to tree networks as shown in Algorithm 1.

Algorithm 1 Time Scheduling Algorithm

Require: Tree T
Ensure: Time schedule $\vec{TS}$
Step 1: Convert collection to distribution problem
The sink distributes one data packet to each node in the network
Step 2: Multi_Linearize tree topology
Convert tree topology T to a multi_line topology:
for $\forall$ sub-tree $B_i \in T$ **do**
$N \leftarrow$ maximum hop count of sub-tree B_i
$L_i \leftarrow$ an N-node line
$L_i(m)$ represents the mth node in L_i, with $P_i(m)$ packets to receive from the sink
$\forall m \leq N$, initialize $P_i(m) = 0$
for $\forall$ node $v \in$ sub-tree B_i **do**
$h \leftarrow$ node v's hop count
$P_i(h) \leftarrow P_i(h) + 1$
end for
end for
Step 3: Scheduling among Multi_line
while sink still has data packets **do**
for $\forall L_i$ that converted from a sub-tree $B_i \in T$ **do**
$t_i \leftarrow$ the last busy slot of L_i by equation (3) based on $\vec{P_i}$
end for
Decide toward which line to transmit: choose the line L_i with the largest t_i while satisfying half-duplex feature
Update time schedule $\vec{TS'}$ for the distribution problem
$Time\_slot++$
end while
Step 4: Mapping back to collection
$\vec{TS} \leftarrow$ mirror $\vec{TS'}$ for the distribution problem to the original collection problem

E. Ordering

The way we have designed and described the algorithms for time and channel scheduling can be implemented in two different orders, resulting in different conflict graph H:

1. Channel Allocation Before Time Scheduling: In this case, H can be defined as follows. If i is interfering with any child of node j, then we place the edge (i, j) in H. This can be determined from the routing tree T, and the interference graph $I(V, E')$.

2. Channel Allocation After Time Scheduling: In this case, we define H as follows. Start with constructing the H as in the previous case. After time scheduling is done, remove the edges (i, j) between nodes i and j if they are not scheduled to receive on any simultaneous slot.

From the definitions, it is clear that the conflict graph for the second approach (channel allocation after time scheduling) is a strict subset of the conflict graph for the first approach. It is possible to see that the number of channels needed in the second approach is smaller than in the first approach. In section VI-D, we will evaluate both orders, and show that the latter, in fact, is more efficient.

F. Lightweight Time Synchronization

Synchronization is crucial for a multi-channel scheduling protocol. We implement a Lightweight Time Synchronization Protocol (LTSP) that is similar to FTSP [19] in that one-way synchronization is performed between every child and its parent on the tree, using MAC-layer time-stamps and linear regression over multiple packets. We also piggy-back other configuration and control information for scheduling and channel allocation on the time-sync packets. During the synchronization phase, all nodes are set to operate on the same channel.

The topology information is first distributed to all nodes in the network. So that a node is able to identify its location and parent in the tree. When the routing configuration is received, a node starts to synchronize with its parent. Meanwhile, it also updates channel configuration and time scheduling. After a child gets synced with its parent, it needs to propagate time synchronization and configuration information to its children if it is not a leaf node. Then the node gets ready for data collection. This process will be repeated periodically in long-term operation.

G. Data Collection

There are 5 major components in the distributed part of MCC implemented at each sensor node, as shown in Fig. 3:

- **Routing Engine**: It obtains routing information, which is generated at the sink through LTSP packets.
- **Forwarding Engine**: It is responsible for maintaining a queue of packets to transmit. The packets could be generated by the node's own application or received from its children. Also provides retransmission mechanism (when ACK is enabled).
- **Channel Controller**: Determines which channel a node uses for transmission and reception.
- **Lightweight Time Synchronizer**: Implements LTSP network synchronization.
- **Time Scheduler**: Maintains the time schedule and time calculations needed to inform the node when to transmit and when to be in receive mode.

V. Overview of Experiments

All our experiments were conducted on the Tutornet indoor testbed (the size of the testbed and more details about the environment is described in [22]), with a set of 30 nodes. We primarily use channels 25 and 26 (IEEE 802.15.4 channels) on

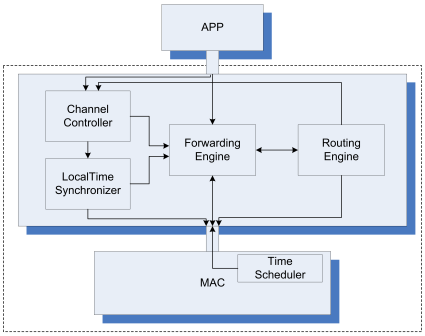

Fig. 3. Software Architecture for distributed part of MCC

the CC2420 radio, which have no overlap with Wi-Fi signals. When needed, we also use channels 11, 24, and 20, which we found empirically to suffer relatively low interference on our testbed. We use a PRR threshold of $\Theta = 90\%$ in all our experiments.

VI. Parameter Evaluation

We evaluate each building block in MCC: time synchronization, routing, channel allocation, and scheduling.

A. Overhead of Time Synchronization

LTSP is responsible for the time synchronization. It uses MAC layer time-stamping and linear regression to provide precision of jiffy-level (1 jiffy $\sim 30.5\mu s$).

To evaluate the performance of LTSP, we test it on a 30-node star-topology. Fig. 4 shows the average synchronization error measured in terms of the residual time drift observed after a round of synchronization. We vary the size of the linear regression table (i.e. number of packets over which the skew and offset are computed). The figure shows that we can reduce the error to below 0.007 jiffies per second on average; this corresponds to a residual drift error about 0.21 microseconds per second, comparable to the best known results in the literature (e.g., on the order of a microsecond per second for FTSP [19]).

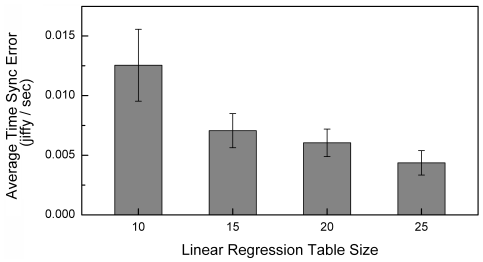

Fig. 4. LTSP: Time Synchronization Accuracy Comparison

We conduct a testbed experiment to understand how frequently the synchronization needs to be repeated. Because in all our experiments the maximum depth of network is 4 hops, we do this experiment with a simple 5-node line topology. We let the protocol run with a single synchronization event before the start of the data collection at time 0. Results showed that, setting aside times when there was heavy external interference, there is a consistent deterioration in the performance starting around 150 minutes after the beginning of network operation. The network synchronization process could be completed in under 2 minutes, implying that the overhead due to synchronization will be typically less than 2%.

B. Generation of a Balanced Routing Tree

Fig. 5 shows the maximum sub-tree size obtained using the CMS tree algorithm on the 30 node testbed for different power settings, averaged over all 30 possible sink locations. We see that even at the lowest power, the average maximum sub-tree size n_k is well below the bound of $(N+1)/2$ (which in our case comes to 15.5). This ensures that the sink will always be the bottle-neck. An ancillary benefit of this algorithm is that it yields a relatively shallow tree, with small maximum hop count, which is beneficial for reducing the synchronization time, delay, packet loss (in the case of no ACK). This is also illustrated in Fig. 5, which shows that the hop-count is always less than 5, and often just 2 hops at the medium to high transmit power settings.

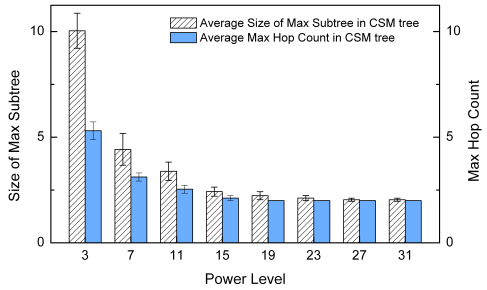

Fig. 5. Maximum sub-tree size and max hop count obtained by CMS Algorithm in 30-node networks

C. Time Scheduling: Slot Length

The main parameter for time scheduling is the length of slots, which should be a function of the packet size. We identify the optimal schedule length for each packet size by conducting experiments where we vary the slot-length and observe its impact on the throughput. We conduct two sets of experiments; one to measure the impact of slot length for a star-topology, to understand the best slot-length for maximizing the rate of sink reception, and another to measure the impact of slot length for a linear topology, to understand the best slot-length for maximizing the relay transmission rate (subject to the MCC design constraint that the reception slots are twice as long as the transmission slots).

A typical set of results, for the 40-byte, no-ACK case is shown in Fig. 6. We see that a smaller slot-length causes packet losses, indicated by the delivery rate falling significantly below 100%, a larger slot-length results in poor utilization. Interestingly, we find that the maximum sink RX rate in the star-topology is achieved at a slot-length of 140 jiffies, while the maximum relay transmission rate in the linear topology achieves a maximum at a slot-length of 160 jiffies. This can be attributed to the additional channel-switching overhead incurred in the linear topology. Optimal slot length was obtained empirically since software delays are hard to estimate precisely.

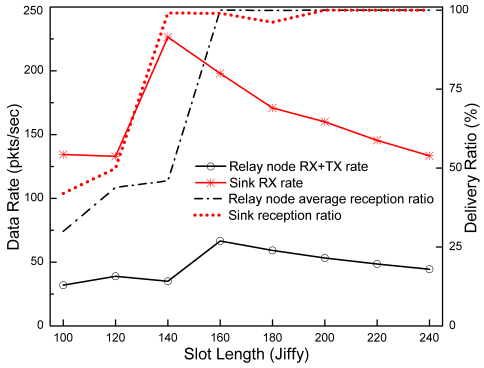

Fig. 6. The impact of time slot length on relay transmission and sink reception (packet size = 40 bytes)

Fig. 7 shows the value of the best slot-length as the packet size is varied, for both the sink and relay nodes. Again, we see that the best slot-length for the relay node is generally higher until a packet size of about 80 bytes. After this point, it appears that the overhead due to channel switching is negligible and the two cases require the same slot-length. In the MCC protocol implementation, we use the higher of the two curves, i.e. the curve for the relay nodes, to set the slot-length for all nodes in the network according to the corresponding packet size[2]. One implication of this plot is that for small packet sizes, the sink will not be fully utilized, resulting in some reduction of the throughput compared to the maximum possible sink RX rate. Note that these plots are both for the case without ACK. Similar curves are obtained in the case ACK is used, however, as expected, the best slot lengths in that case are higher.

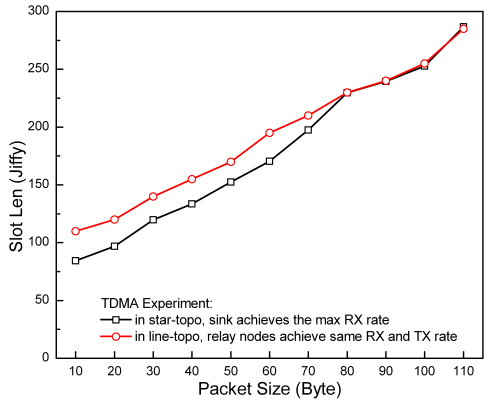

Fig. 7. Optimal slot lengths for sink and relay nodes in MCC

D. Channel Allocation

As mentioned in section IV-E, the order of channel allocation and time scheduling algorithms can be chosen in both ways. Using the connectivity obtained from the testbed at different power levels, we compare the number of channels required for the two approaches, after applying the CMS algorithm to determine the balanced routing tree, for each of the 30 possible sink locations. This is shown in figure 8 using box plots showing the minimum, 25 percentile, median, 75 percentile, and maximum values as well as the mean (shown as a small square) of the number of channels required in each case. It can be seen from this evaluation that the approach of doing channel allocation *after* time scheduling is decidedly better: in most cases, only 2 channels are required, at most 4 in the low power settings. Channel minimization is an important consideration because even if the number of required channels is less than the 16 available on the radio, many channels may show poor quality in indoor environments due to WiFi interference. For the remaining results, we therefore always run channel allocation after time scheduling.

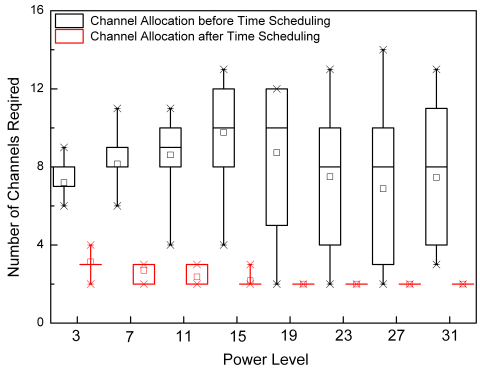

Fig. 8. Number of channels required to eliminate interference

VII. MCC Performance Evaluation

In this section, we evaluate MCC collection under different settings to see how factors such as packet size, power level, retransmission mechanism (ACK or no-ACK) and channel allocation impact the performance of the network, and compare it with the estimated maximum achievable throughput, as well as with other protocols.

A. Backlogged MCC

MCC collection maintains a queue of packets, from both its own applications and the network. When a node is scheduled to transmit, if this queue if not empty, it will send the first packet; otherwise, the current TX slot is wasted. We first test MCC collection throughput when the queue is always backlogged so that all TX slots are busy and utilization is 100%.

We perform the test on a 10-node and a 30-node balanced tree on the testbed. We show the topology of the first case in Fig. 9. Maximum power level (31) is used to have the best link quality. We label in Fig. 9 the scheduled TX slots of each node. Note that all these slot numbers are relative to its parent's first TX slot. The total frame length is 9. We test MCC collection throughput with packet sizes of 40 bytes and 100 bytes, and with ACK enabled and disabled. When ACK is enabled, at most 3 retransmissions are executed.

The result is shown in Fig. 10. We compare MCC collection throughput with the expected throughput (obtained in section III) with and without ACK. We see that for the 100-byte packet without ACK, the throughput of MCC collection is $\sim 99kbps$, almost achieving the maximum achievable RX capacity. However, for the 40-byte packet, we see that the throughput is a bit lower than the estimated maximum sink

[2]We assume in our evaluations, as in most WSN applications, that packets are all of the same length; in the rare case of an application where multiple packet sizes are utilized, the slot-length should be chosen to correspond to the maximum packet size, though this may result in lower utilization.

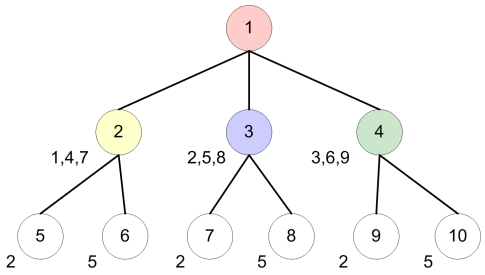

Fig. 9. A balanced routing tree with 10 nodes

RX rate. This is due to the channel-switching overhead which causes the relay nodes to require a longer slot-length than the sink-rate-maximizing slot-length. A similar trend is observed for backlogged MCC with ACK. This figure shows that MCC is able to achieve close to the maximum achievable throughput so long as nodes always have data to send at each slot.

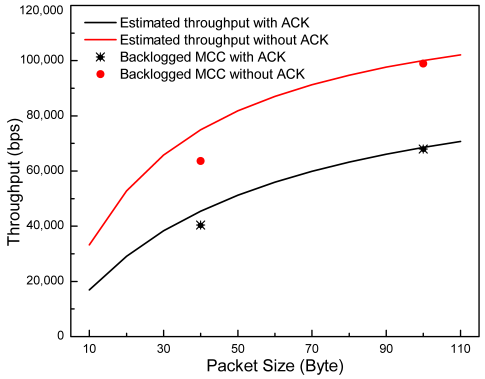

Fig. 10. Throughput of backlogged MCC compared with the expected maximal achievable throughput with or without ACK

B. Comparison with other protocol

In Sec. VII-A, we bypassed the queue management. Now we assume the application generates a constant data rate equal to the maximum that can be transported by the MCC protocol.

We deploy MCC collection on the full 30-node testbed. We consider three power levels. For a fair comparison, we enable ACK and test for packet size of 40 bytes and 100 bytes. MCC is compared with two alternatives: CTP and RCRT, a state-of-the-art centralized rate controlled collection protocol that also guarantees end to end reliable transport. CTP protocol is implemented with an ideal maximum achievable rate, which is calculated by extensively tests considering all fair rate allocation for all nodes.

While MCC and CTP do not provide for end to end reliability, they provide hop-by-hop reliability through the use of ARQ and retransmissions. Our choice of RCRT as a comparison collection protocol is motivated by the study in [7] which shows that this centralized approach provides higher throughput than the best distributed rate control protocols that works over CTP (namely, IFRC [4]). The throughput comparison resulting from our experiments is shown in Fig. 11.

The experiments show that for all schemes, as expected, the throughput is higher for larger-sized packets. MCC consistently offers in all cases the highest throughput. We find

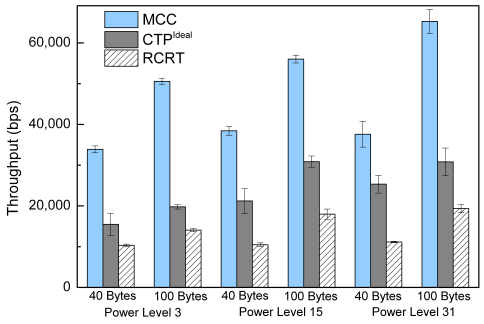

Fig. 11. Network throughput comparison of MCC, CTP-Ideal, and RCRT in a 30-node network

that with respect to CTP the relative throughput gain of MCC is the lowest for 40-byte packets at the highest power level (48% improvement), and is the highest for 100-byte packets at the lowest power level (155% improvement). With respect to RCRT, the throughput gain of MCC is much higher and varies somewhat less across all settings (from 227% to 266%).

We emphasize that the purpose of this experiment is not to claim that MCC is a better protocol *per se* than CTP or RCRT from all perspectives. RCRT, for instance, provides the additional key functionality of end-to-end reliability, which may incur a throughput penalty. These results give an indication of how much additional throughput improvement can be obtained with a near-optimal joint routing-TDMA-FDMA approach.

C. Scaling

We examine the scaling of MCC in networks with different sizes ranging from 10 to 30 as shown in Fig. 12. We choose the network in these experiments as a monotonically increasing set, i.e. the network used in the experiment for size m is a subset of the network used in the experiment for size n, all $m < n$. The experiments are for packet size of 40 bytes and the power level of 15. Throughput and fairness (using Jain's index) are measured and compared with what is obtained with CTP with the ideal rate allocation. For MCC, we found that in all cases the routing algorithm produced a balanced tree, and since a conflict-free channel allocation is always possible, the time slot allocation fully occupied the sink in all cases. Hence the total source transmission rate was the same in all cases. However, in the larger networks, due to an increase in hop-count and weak links, we do see a mild deterioration in throughput. This suggests that the scaling of MCC performance with respect to network size is graceful on the whole, even for network size greater than 30 nodes. And fairness also remains high. CTP with ideal fair rate allocation shows a consistently lower throughput, and a slightly higher rate of deterioration at larger network sizes, resulting from increased interference and contention on the single channel.

VIII. Discussion and Future Work

This work has made advances in identifying a practical approach to maximize the throughput of data collection in WSN, but there are other dimensions in which it can be extended.

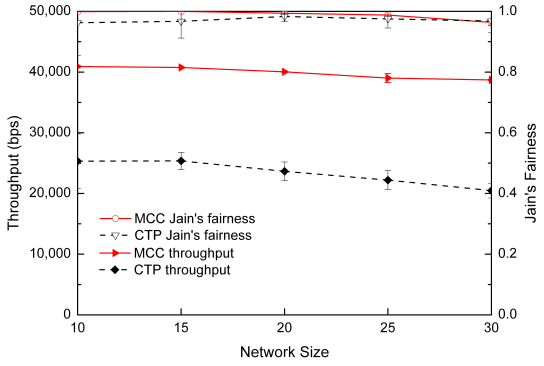

Fig. 12. Network throughput and Jain's fairness

- **Dynamics:** An inherent weakness of centralized scheduling is that they are not geared towards handling dynamics due to time-varying links, node joins and failures, or frequent traffic changes. While one approach is to repeat the configuration phase from scratch periodically, there is a trade-off between the overhead and agile response. The use of frequency hopping (FH) techniques can also improve the reliability of the network in face of dynamic environments. As stated in section II, the extension of MCC to support FH is straightforward and would help to strength the protocol against sporadic noise and interference.
- **Channel Heterogeneity and Availability:** In this work we have assumed that all available channels have the same quality and are symmetrical. Also, we assumed that there are a sufficient number of channels available to be used. Currently we are working on the collection of link quality on all available channels and on new channel and time scheduling algorithms which can handle heterogeneous environments, and take into account asymmetric links.
- **Energy Consumption:** Energy consumption is an important issue in WSN that was not yet explored by MCC. Due to the TDMA feature, each node is aware of its TX/RX time schedule and it is easy to configured the radio to be turned off during idle periods. Using the energy consumption numbers for reception and transmission of the CC2420 radio, we analysed the 30-node network considered in the evaluation of MCC and found that we can yield energy reduction ranging from 85 to 95%, simply by turning off the radios.

IX. Conclusion

In this work, we have evaluated the maximum sink RX rate with respect to key parameters such as packet size, power level, acknowledgements. We have shown experimentally that by having a load balanced tree, multiple channels and time scheduling, it is possible to achieve a throughput that is close to this maximum sink RX rate. We have shown how to make this approach efficient, in terms of minimizing the channels required, by choosing channels based on time scheduling information. We have shown that this approach yields significant benefits in terms of throughput, at least in static environments, over state of the art single-channel collection protocols based on random access. As discussed above, future work could extend these ideas to dynamic environments.

References

[1] G. Werner-Allen, K. Lorincz, M. Welsh, O. Marcillo, J. Johnson, M. Ruiz, and J. Lees, "Deploying a Wireless Sensor Network on an Active Volcano", *IEEE Internet Computing, Mar/Apr 2006*.

[2] M. Bathula, M. Ramezanali, I. Pradhan, N. Patel, J. Gotschall, and N. Sridhar, "A sensor network system for measuring traffic in short-term construction work zones", *IEEE DCOSS 2009*.

[3] A. Sridharan and B. Krishnamachari, "Explicit and Precise Rate Control for Wireless Sensor Networks", *ACM SenSys 2009*.

[4] S. Rangwala, R. Gummadi, R. Govindan and K. Psounis, "Interference-Aware Fair Rate Control in Wireless Sensor Networks", *ACM SIGCOMM 2006*.

[5] S. Moeller, A. Sridharan, B. Krishnamachari, and O. Gnawali, "Routing Without Routes: The Backpressure Collection Protocol", *ACM/IEEE IPSN 2010*.

[6] S. Kim, R. Fonseca, P. Dutta, A. Tavakoli, D. Culler, P. Levis, S. Shenker, and I. Stoica, "Flush: A Reliable Bulk Transport Protocol for Multihop Wireless Networks", *ACM SenSys 2007*.

[7] J. Paek, R. Govindan, "RCRT: Rate-Controlled Reliable Transport for Wireless Sensor Networks", *ACM SenSys 2007*.

[8] O. Gnawali, R. Fonseca, K. Jamieson, D. Moss, and P. Levis, "Collection Tree Protocol", *ACM SenSys 2009*.

[9] O. D. Incel, A. Ghosh, B. Krishnamachari, and K. Chintalapudi, "Fast Data Collection in Tree-Based Wireless Sensor Networks", *IEEE Transactions on Mobile Computing, 2011*.

[10] F. Österlind and A. Dunkels, "Approaching the maximum 802.15.4 multi-hop throughput," *Proceedings of the Fifth ACM Workshop on Embedded Networked Sensors (HotEmNets 2008)*, June 2008.

[11] O. D. Incel, *Multi-Channel Wireless Sensor Networks: Protocols, Design and Evaluation*, Ph.D. Thesis, University of Twente, Netherlands, 2009.

[12] G. Zhou, C. Huang, T. Yan, T. He, and J. A. Stankovic, "MMSN: Multi-Frequency Media Access Control for Wireless Sensor Networks", *IEEE INFOCOM 2006*.

[13] Y. Wu, J. A. Stankovic, T. He, J. Lu, and S. Lin, "Realistic and Efficient Multi-Channel Communications in Wireless Sensor Networks", *IEEE INFOCOM 2008*.

[14] H. Le, D. Henriksson, and T. Abdelzaher, "A Practical Multi-Channel Medium Access Control Protocol for Wireless Sensor Networks", *ACM/IEEE IPSN 2008*.

[15] Doddavenkatappa, Manjunath and Chan, Mun Choon, "P^3: A Practical Packet Pipeline Using Synchronous Transmissions for Wireless Sensor Networks", *ACM/IEEE IPSN 2014*.

[16] B. Raman, K. Chebrolu, S. Bijwe, V. Gabale, "PIP: A Connection-Oriented, Multi-Hop, Multi-Channel TDMA-based MAC for High Throughput Bulk Transfer", *ACM Sensys 2010*.

[17] Y. Kim, H. Shin, and H. Cha, "Y-MAC: An Energy-Efficient Multi-channel MAC Protocol for Dense Wireless Sensor Networks", *ACM/IEEE IPSN 2008*.

[18] C. H. Papadimitriou, "The complexity of the capacitated tree problem", *Networks, vol. 8, no. 3, 1978*.

[19] M. Maroti, B. Kusy, G. Simon, and A. Ledeczi, "The flooding time synchronization protocol", *ACM SenSys 2004*.

[20] C. Florens, M. Franceschetti, and R.J. McEliece, "Lower bounds on data collection time in sensory networks", *IEEE Journal on Selected Areas in Communications, 22 (6) 2004*.

[21] D. J. A. Welsh, and M. B. Powell, "An upper bound for the chromatic number of a graph and its application to timetabling problems", *The Computer Journal, vol. 10, no. 1, 1967*.

[22] Tutornet: A Tiered Wireless Sensor Network Testbed - *http://anrg.usc.edu/www/tutornet/*

[23] K. S. J. Pister, L. Doherty, and H. Ave, "TSMP: Time Synchronized Mesh Protocol", *IASTED International Symposium Distributed Sensor Networks, 2008*.