WIP: BurstMAC — A MAC Protocol with Low Idle Overhead and High Throughput

Matthias Ringwald, Kay Römer
Institute for Pervasive Computing, ETH Zurich, Zurich, Switzerland
Email: {mringwal,roemer}@inf.ethz.ch

***Abstract*— Many sensor network applications feature bursty traffic patterns: after long periods of idle time with almost no network traffic, large amounts of data have to be transmitted reliably and in a timely manner. One example is volcano monitoring [8], where precious high-volume data is generated by rare volcanic eruptions.**

Unfortunately, existing MAC protocols do not sufficiently support such applications with bursty traffic patterns. Protocols design for low data rates such as WiseMAC or SCP-MAP have very low overhead in idle situations, but have high overhead and low throughput under high load due to collisions. In contrast, scheduled protocols such as LMAC can handle high loads without collision, but have low throughput and significant overhead in idle mode [4].

We devise a new MAC protocol, BurstMAC, that closes this gap by combining low idle overhead with high throughput under load. It achieves radio duty cycles of $< 1\%$ in an idle network and uses up to 71% of the available bandwidth during traffic bursts without any collisions. [1]

I. Protocol Overview

BurstMAC combines a number of techniques to combine high throughput under load with low idle overhead. Most notably, scheduling and the use of multiple radio channels enable high throughput, while cooperative transmissions and techniques to eliminate preambles guarantee low idle overhead. In this section, we present the key ideas behind BurstMAC and outline the basic protocol structure

A. Collision-free Communication

To avoid collisions, BurstMAC operates in synchronous rounds. The sink node is used as time reference for synchronization. Each node synchronizes to the average time of all nodes which are closer to the time reference than itself. Each round consists of 32 frames. Every frame contains a control section and a data section as depicted in Fig. 1. To maximize throughput and to allow for collision-free communication during the data section, BurstMAC uses 32 interference-free data channels and one control channel. The control section is used for time synchronization, to broadcast other information to all network neighbors, and to assign color ids to nodes. As a result of the latter, each node is assigned a color id $c \in 1..32$ that is unique within two hops. The color id c is used for two purposes. Firstly, the control section of frame c is reserved for the node with color id c, which allows a node to send control messages without collision on the control channel in frame c. Secondly, the node receives data on radio channel c during the data section, coordinating multiple senders to receive data without collisions.

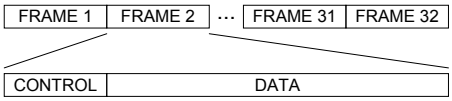

Fig. 1. BurstMAC round consisting of 32 frames which each contain a CONTROL and a DATA section.

B. Coordination-free Transmission Scheduling

As a node cannot send and receive at the same time, some form of coordination is required among the nodes to achieve agreement on when to send and when to receive. To realize this coordination without introducing additional control messages, BurstMAC uses the following approach. During each frame, a node is either in transmit or receive mode, that is, it can either only transmit or only receive data during the whole frame. The choice of mode is controlled by a pseudo-random number sequence which is seeded with the unique 16-bit node id. Knowing the node ids of its neighbors, a node can not only compute its own current mode, but also the current modes of its neighbors. If node A wants to send to neighbor B, then A has to wait for a frame when it is in send mode and B is in receive mode. A uses B's channel for the actual transmission. This approach avoids any extra traffic for coordination among nodes.

C. Cooperative and Single-Bit Transmission

These two physical layer techniques allow for efficient scheduling in the data section of BurstMAC. If multiple senders send a jamming signal at the same time, a receiver can use the Received Signal Strength Indicator (RSSI) to detect that at least one node is sending. This *cooperative transmission* is used to quickly detect if at least one sender wants to send a packet.

To query which nodes want to send, *single-bit transmission* is employed. For this, a coordinating node broadcasts a short synchronization packet to provide a bit-accurate time reference. If node c wants to send a packet, it sends a jamming signal in bit slot c which is detected by the coordinating node. Both techniques are described in more detail in [6].

[1]The work presented in this work-in-progress paper was partially supported by the Swiss National Science Foundation under grant number 5005-67322 (NCCR-MICS)

D. Packet Bursts

To increases throughput and reduce communication overhead, a sender can request the transmission of multiple packets in a row, eliminating lengthy preambles for all but the first packet. Still, each packet has an individual checksum to detect bit errors. The receiver replies a bit vector with a one bit for each packet that has been received correctly.

E. Cross-layer Optimizations

Typical routing protocols such as MintRoute [9] need to perform neighbor discovery and link quality estimation, which requires each node to broadcast beacon packets at regular intervals. However, due to the existence of the control packets in BurstMAC, we can integrate neighbor discovery and link estimation into BurstMAC without additional overhead.

II. Protocol Details

A. 2-Hop Coloring

Each node has to be assigned a color id c which is unique within two hops. As discussed in Sect. I, c is used as a channel id for payload data transmissions and to schedule broadcasting of control messages on the control channel.

For coloring, all nodes keep track of the frames used by their neighbors for sending control messages similar to LMAC [7]. As a node with color id c transmits a control message in frame c, each node is aware of the colors assigned to its neighbors and periodically broadcasts a bit vector of these occupied color ids in its control message (field `occupied`). A newly joining and yet uncolored node with id i receives the list of used color ids in the control messages of all of its neighbors. The union of these sets equals the set of color ids used in its 2-hop neighborhood. The new node then randomly picks a color id c from the remaining free colors and transmits its control packet in frame c in the next round. A special flag in the control message requests other nodes to echo the node id contained in the control packet of frame c. If another node with node id j simultaneously picks the same color c, both node ids i and j will be reported for frame c by different neighbors. In this case, both newly colored nodes pick another free color at random.

B. Transmission Scheduling

BurstMAC efficiently schedules data transfer on demand using cooperative and single-bit transmission. Fig. 2 shows the data section in more detail, time increases from left to right. One node is in receive mode and two nodes in send mode want to transmit a packet to the receiver. In segment A, both senders employ cooperative transmission and concurrently transmit their send request to the receiver. If there is at least one sender, the receiver exerts the single-bit transmission technique by sending a minimal sync packet in segment B. The sync packet allows a sender to accurately synchronize and send a single jamming "bit" in the slot which corresponds to its color id c in segment C. Based on the list of senders, the receiver than computes and broadcasts the transmission schedule in segment D. In each data segment E_x, a sender transmits a data

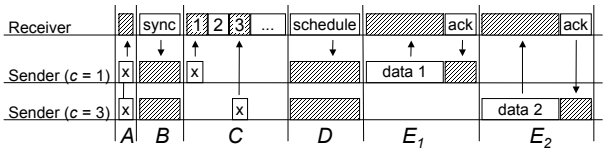

Fig. 2. DATA section with two senders transmitting a packet to the receiver. Cooperative transmission is used in segment A and single bit transmission in segments B and C to identify the senders.

packet which is immediately acknowledged by the receiver. The header of the data message also contains several flags. One of them, the `more` flag, can be set by the sender to indicate that it needs another slot to send a further packet. If available, the receiver will reply an unused slot k in the acknowledgement, such that the sender can send the next packet in slot E_k.

C. Network Startup

All nodes concurrently send a short jamming signal, called *Blip*, for 100 us at the very beginning of the control section on an extra Blip channel. By this, a new node joining the network has to scan the control channel at 100% duty-cycle only for a single frame instead of a whole round. On detection of the Blip, the node already has approximate timing information on the start of the control section and will receive at least one control message in the next 32 frames.

III. Implementation

We implemented BurstMAC on BTnode Rev. 3 nodes [1]. They basically consist of an ATMEL ATMEGA128 8-bit microcontroller, 256 KB of external SRAM, a ChipCon CC1000 radio module and a Zeevo ZV-4002 Bluetooth module. The Bluetooth module was not used in this work. BTnut, an extension of Nut/OS [3], is used as the operating system.

The CC1000 on the BTnode is configured for the 868 MHz ISM-band, where the actual baseband frequency within this band is configurable by software. We used 34 channels of which one is reserved for control broadcasts, one is used for the wake-up mechanism and the other 32 are used for data communication. The analog RSSI output of the CC1000 is used for clear-channel assessment, cooperative and single-bit transmission. We further make use of the CC1000's ability for precise MAC layer timestamping in the order of 10 us [5].

For the implementation, we used a bit rate of 19200 bps and a frame length of 1 s, which is split into 50 ms for the control section and 950 ms for the data section. By this, the data section provides 24 data slots for single bit transmission or up to 51 packets in burst mode. A packet contains a type field and up to 32 bytes of payload. In this configuration, a node can send or receive a single packet and up to 52 burst packets of each 33 bytes in a single frame. Such a transfer of 1716 bytes results in a maximal usable bandwidth of 71.5% of the total bandwidth of 2400 bytes/s.

IV. Preliminary Evaluation

We study the performance of the BurstMAC implementation on BTnodes. In particular, we investigate the accuracy of time synchronization, the idle overhead, as well as overhead and time to completion of traffic bursts.

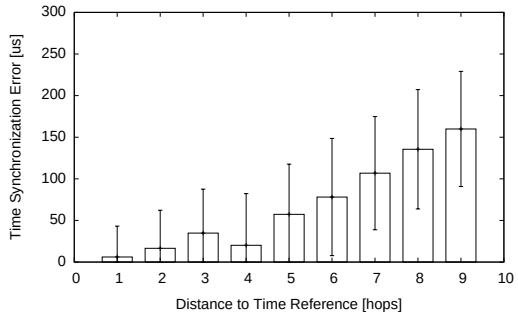

Fig. 3. Accuracy of time synchronization on a chain topology.

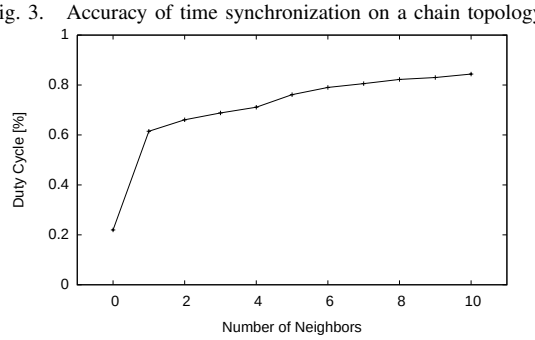

Fig. 4. Radio duty cycle in idle mode as a function of neighborhood size.

A. Time Synchronization

We investigate the accuracy of time synchronization as a function of the network diameter. For this, we arrange 10 nodes in a chain topology with the time reference at the end, forcing each node to use only its direct parent as a reference for synchronization. However, all nodes are within communication range of the time reference, such that each node can directly measure its synchronization error with respect to the time reference. We ran this setup for one hour, where each node measures its synchronization error once per round. After collecting the measurements from the network, we computed averages and standard deviations for all nodes in the chain as depicted in Fig. 3. The results show that the average synchronization error increases by about 25 μs per hop. Note, however, that the accuracy of our clock is only 30.5 μs. As the required synchronization accuracy is in the order of few milliseconds (we only need to synchronize to rounds, not at the byte or bit level), we can easily support networks with a diameter in the order of several tens of hops.

B. Idle Case

We investigate the idle overhead of our protocol in terms of the radio duty cycle. As the radio duty cycle is a function of the number of neighbors of a node (a node has to receive the control message from each of its neighbors), we study the radio duty cycle of a node with a varying number of neighbors. For each neighborhood size, we ran the network for 5 minutes, measuring the time the radio was on and compute the average duty cycle as depicted in Fig. 4. We find that the duty cycle increases by about 0.02 % per added neighbor. For zero neighbors, the duty cycle of the low-power listening used during the network startup is measured.

C. Burst Case

To investigate BurstMAC's performance in the burst case, we setup a network of 30 nodes in our lab. The diameter of the resulting network is 4 hops and each node has at most 4 neighbors.

We simulate a traffic burst in the volcano monitoring application [8], where an eruption triggers all nodes simultaneously to transmit a burst of 10 KB (i.e., 320 BurstMAC packets). We measure the following two metrics. Firstly, the time it takes until all data has been successfully delivered to the sink. Secondly, the average overhead per transmitted packet. For the latter, we measure the average radio-on time per packet and put this value into relation to the minimal radio-on time it takes to transmit a packet (i.e., packet length divided by radio bitrate). BurstMAC delivered the total of 9280 packets in average time of 448 seconds which means that 20.7 packets could be received by the sink per second. This is only half the maximal packet reception rate of 51 packets per second using the packet burst. Further investigation has to evaluate, if this throughput can be improved.

To estimate the protocol overhead, we calculated the total radio-on time based on the collected metrics. In addition, we calculated the optimal radio-on time based on a packet length of 35 bytes (1 byte type, 32 bytes payload and 2 byte CRC) to be 14.5 ms. As the transmission of a packet involves both the sender and the receiver, we define the optimal radio on-time as 29 ms. BurstMAC required an average radio time of 46 ms which is 58% more than the optimal radio time and this already includes the control messages and the overhead caused by sending preambles.

V. Outlook

We are currently working on a thorough evaluation using 30 BTnodes distributed in our lab which involves comparison to other protocol such as LMAC and SCP-MAC. Support for very dense networks is a further topic as 32 channels might not be sufficient for the 2-hop-coloring of very dense networks.

References

[1] BTnodes. A distributed environment for prototyping ad hoc networks. www.btnode.ethz.ch.

[2] Nicolas Burri, Pascal von Rickenbach, and Roger Wattenhofer. Dozer: ultra-low power data gathering in sensor networks. In *IPSN 2007*, pages 450–459, April 2007.

[3] ethernut. Embedded ethernet. www.ethernut.de/.

[4] Koen Langendoen. Medium access control in wireless sensor networks. In H. Wu and Y.Pan, editors, *Medium access control in wireless networks, volume II: practive and standards*. Nova Science Publishers, 2007.

[5] Miklós Maróti, Branislav Kusy, Gyula Simon, and Ákos Lédeczi. The flooding time synchronization protocol. In *SenSys*, 2004.

[6] Matthias Ringwald and Kay Römer. Bitmac: A deterministic, collision-free, and robust mac protocol for sensor networks. In *EWSN*, 2005.

[7] L. van Hoesel and P. Havinga. A Lightweight Medium Access Protocol (LMAC) for Wireless Sensor Networks: Reducing Preamble Transmissions and Transceiver State Switches. In *INSS 2004*, Tokyo, Japan, June 2004.

[8] Geoff Werner-Allen, Konrad Lorincz, Jeff Johnson, Jonathan Lees, and Matt Welsh. Fidelity and yield in a volcano monitoring sensor network. In *OSDI*, 2006.

[9] Alec Woo, Terence Tong, and David Culler. Taming the underlying challenges of reliable multihop routing in sensor networks. In *SenSys*, 2003.