Competition: Aggressive Synchronous Transmissions with In-network Processing for Dependable All-to-All Communication

Beshr Al Nahas, Olaf Landsiedel
Department of Computer Science and Engineering
Chalmers University of Technology, Sweden
beshr, olafl @chalmers.se

Abstract

Low-power wireless networking needs to survive interference in order to accommodate the requirements of serious applications of the Internet of Things. Synchronous transmission techniques like Glossy and Chaos perform well under normal operating conditions. However, their data delivery latency suffers under interference even when extended to use channel hopping. In this paper, we discuss our techniques to enhance the robustness of synchronous flooding while keeping the latency and power consumption minimal and how to collect data from multiple sources to multiple destinations.

1 Introduction

Dependable low-power wireless protocols are a key enabler for serious applications of the Internet of Things. Data is too valuable to be lost, timeless delivery is at least favorable and low-power operation is crucial for battery-driven applications. However, interference from ubiquitous wireless devices or from malicious agents deeply challenges the low-power wireless communication as it causes data loss, increased latency, and wasted energy.

In this paper, we try to incorporate what we learned from our previous participation in the competition [2], and we present a simple protocol that tries to achieve and maintain a low-latency and low-power communication scheme that is robust against interference.

Outline. We first present A^2, the synchronous transmissions protocol we base our solution on in §*2*, then we discuss frequency diversity and data redundancy techniques to increase robustness against interference in §*4*, and we conclude in §*5*.

2 Background: A^2 and Synchrotron

A^2 [1] builds on top a synchronous transmissions kernel, Synchrotron, and utilizes in-network processing to provide primitives for network-wide, all-to-all dissemination, collection, aggregation, voting, consensus (two- and three-phase commit) and membership services. A^2 operates in *rounds* where nodes send packets synchronously and receive data thanks to the *capture effect*. In contrast to Glossy and LWB, nodes in A^2 synchronously send different data. At each reception, nodes apply a user-defined in-network aggregation function before sending again in the next *slot*, until completion of the round.

Synchrotron: Synchronous transmissions and capture effect. Synchrotron roots in approaches to synchronous transmissions, such as Chaos, where multiple nodes synchronously transmit the data they want to share. Nodes overhearing the concurrent transmissions receive one of them with high probability, due to the capture effect. For example, to achieve capture with IEEE 802.15.4 radios, nodes need to start transmitting within the duration of the preamble of 160μs [7]. Typically, in 802.15.4, the radio receives the stronger one of the synchronous transmissions if its signal is 3 dBm stronger than the sum of other signals, the so-called *co-channel rejection*.

Synchrotron operates as a time-slotted protocol. The minimum time unit is a *slot*, which fits one packet transmission or reception and processing. Slots are grouped in *rounds*, where a designated function, such as collect or disseminate is run network-wide. Within each slot, a node transmits, receives or sleeps according to the transmission policy of the application.

In-network aggregation. In A^2, each packet contains so-called *progress flags*, where one bit is assigned to each node in the network. The coordinator node starts an A^2 *round* by sending a packet with only its own flag set. Upon successful reception, a node sets its flag and merges the received packet with its own. It transmits in the next time slot when it receives new information, *i.e.*, new flags, or when it sees that a neighboring node is transmitting messages with fewer flags set, *i.e.*, a neighbor knows less than the node itself. The process continues until all nodes have set their flags.

Similar to Chaos, the rules for merging are application specific. A^2 provides merging rules for network-wide dissemination, aggregation, and data collection. With the *OR* operation, for example, A^2 identifies the logical OR value of different inputs: Next to the flags, the only payload is the OR result collected so far. Upon reception, nodes do logi-

International Conference on Embedded Wireless Systems and Networks (EWSN) 2018
14–16 February, Madrid, Spain

ISBN: 978-0-9949886-2-1

cal OR between their local value and the payload, write it to the packet payload, merge the flags, and set their flag before transmitting in the next time-slot.

3 Design

We employ A^2 as a basis, and we use the aggregation function OR to collect the different readings of the sources GPIO. Since A^2 is based on flooding, the result will be distributed to all nodes; thus, the respective destinations can make use of the bits of the payload they are interested in.

We run A^2 with a static list of nodes instead of running special join rounds since the source and destination nodes are known beforehand and do not change across runs.

4 Surviving Interference

Both Chaos and Glossy see their performance degrade in presence of interference [4, 6, 5, 8, 3, 2]. We address this by employing the frequency agility and parallel channels features in Synchrotron.

Channel Hopping for Robustness. Nodes in Synchrotron transmit in each time slot on a different frequency following a network-wide schedule, similar to WirelessHART, TSCH or Bluetooth. This prohibits interference on individual channels from disturbing the operation of Synchrotron [9] and allows A^2 to co-exist with other wireless technologies such as 802.11 or TSCH. Moreover, note that synchronous transmissions in combination with channel hopping have shown their robustness during the EWSN dependability competitions, where, for example, all three top-ranking teams in 2017 combine these two design elements [8, 3, 2].

Parallel Channels. The probability of capture reduces as the number of concurrent transmitters increases [7]. To combat this, Chaos proposes reducing the transmission power while in Synchrotron, we use multiple channels in parallel instead. Each node randomly chooses one channel per timeslot to either receive or transmit. The channel is chosen from a shared hopping sequence, and the number of parallel channels is configurable. This directly translates to a chance of a node meeting a neighbor on a given channel. As a result, Synchrotron practically reduces network density and thereby increases the probability of capture. Moreover, it does so without increasing the network diameter, as for example, decreasing the transmission power does. As a side effect, this also increases frequency diversity and robustness further.

Retransmissions. Frequency agility features help improving reliability, but packet losses can still occur if all available channels are blocked while we do a communication round. To alleviate this, we have two levels of retransmission: i) A^2 default transmission policy has embedded retransmissions which we extend to retransmit several times whenever a node learns new information; and ii) we repeat the A^2 communication rounds until we get the acknowledgment flags set or a new event happens. The number of retransmissions represents a tradeoff between communication cost and reliability.

5 Conclusion

In this paper, we present a possible solution to the problem using A^2. Adaptations and tuning of the protocol parameters will be conducted based on experimentation on the target environment.

6 Acknowledgments

We would like to thank the Institute of Technical Informatics at the Graz University of Technology, Austria; especially, Carlo Alberto Boano and Markus Schuß for coordinating the dependability competition. This work was supported by the Swedish Research Council (VR) through the project *ChaosNet: Distributed Computing for Low-Power Wireless Networks* and the Swedish Foundation for Strategic Research (SSF) through the project LoWi.

7 References

[1] B. Al Nahas, S. Duquennoy, and O. Landsiedel. Network-wide Consensus Utilizing the Capture Effect in Low-power Wireless Networks. In *Proceedings of the Conference on Embedded Networked Sensor Systems (ACM SenSys)*, 2017.

[2] B. Al Nahas and O. Landsiedel. Competition: Towards low-power wireless networking that survives interference with minimal latency. In *Proceedings of the European Conference on Wireless Sensor Networks (EWSN)*, 2017.

[3] A. Escobar, J. Garcia, F. Cruz, J. Klaue, A. Corona, and D. Tati. Competition: Redfixhop with channel hopping. In *Proceedings of the European Conference on Wireless Sensor Networks (EWSN)*, 2017.

[4] F. Ferrari, M. Zimmerling, L. Mottola, and L. Thiele. Low-Power Wireless Bus. In *Proceedings of the Conference on Embedded Networked Sensor Systems (ACM SenSys)*, 2012.

[5] Z. He, K. Hewage, and T. Voigt. Arpeggio: A penetration attack on glossy networks. In *Proceedings of the Conference on Sensor, Mesh and Ad Hoc Communications and Networks (IEEE SECON)*, 2016.

[6] K. Hewage, S. Raza, and T. Voigt. An experimental study of attacks on the availability of glossy. *Computers & Electrical Engineering*, 41, 2015.

[7] O. Landsiedel, F. Ferrari, and M. Zimmerling. Chaos: Versatile and Efficient All-to-All Data Sharing and In-Network Processing at Scale. In *Proceedings of the Conference on Embedded Networked Sensor Systems (ACM SenSys)*, 2013.

[8] R. Lim, R. D. Forno, F. Sutton, and L. Thiele. Competition: Robust flooding using back-to-back synchronous transmissions with channel-hopping. In *Proceedings of the European Conference on Wireless Sensor Networks (EWSN)*, 2017.

[9] T. Watteyne, A. Mehta, and K. Pister. Reliability Through Frequency Diversity: Why Channel Hopping Makes Sense. In *Proceedings of the ACM Symposium on Performance Evaluation of Wireless Ad Hoc, Sensor, and Ubiquitous Networks (PE-WASUN)*, 2009.