Competition: Towards Low-Latency, Low-Power Wireless Networking under Interference

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

Abstract

Low latency reliable data delivery is a key challenge for mission critical applications in the Industrial Internet of Things. Chaos was shown to perform well under normal operating conditions. In this paper we discuss some techniques to enhance its robustness under strong interference.

1 Introduction

Reliable data delivery, especially in harsh environments with strong interference, is a key challenge for mission critical applications in Cyber-Physical Systems and the Industrial Internet of Things. These applications, such as wireless control loops, demand low-latency data delivery and high reliability.

Conventional approaches use data collection protocols, for example, CTP [6] or RPL [11]. Both are best-effort protocols and trade latency for reliability; achieving an average latency in the order of seconds in multi-hop deployments [1, 2, 9]. However, once the topology changes, for example, due to interference, RPL and CTP need to re-build their routes, increasing latency, jitter and often reducing reliability. Our approach, Chaos [8], departs from this traditional tree-based routing and bases on synchronous transmissions and distributed data aggregation.

In this paper we present our first results towards a robust version of Chaos, designed for challenging environments with strong interference. We briefly provide in Section 2 the required background on Chaos. Next, we discuss in Section 3 which design aspects of Chaos show limitations under interference. We introduce our extensions for robust networking under inference in Section 4. In Section 5, we discuss adaptations to Chaos to match the application requirements of the competition: data collection towards a sink from a single source, with actuation. Next, we discuss limitations of the design of our Robust Chaos in Section 6 and conclude in Section 7.

International Conference on Embedded Wireless Systems and Networks (EWSN) 2016
15–17 February, Graz, Austria

ISBN: 978-0-9949886-0-7

2 Chaos in A Nutshell

Chaos is an all-to-all data sharing primitive for low-power wireless networks. Unlike current approaches, Chaos essentially parallelizes collection, processing, and dissemination inside the network by building on two main mechanisms: Synchronous transmission and user-defined merge operators.

In Chaos, nodes synchronously send the data they want to share. Nodes overhearing these transmissions receive packets with a high probability due to the capture effect [10]. Upon reception, nodes merge the received data with their own and transmit the results again synchronously. Merging of data happens according to a user-defined merge operator. Chaos allows users to freely program various merge operators, from simple aggregates to complex computations. For example, Chaos computes simple aggregates, such as the maximum, in a 100- node multi-hop network within less than 90 milliseconds. The whole process is triggered by an appointed node, but continues in a fully distributed manner until all nodes in the network share the same data.

3 Chaos: Robust by Design

Chaos and related approaches such as LWB [3] and Glossy [4] rely on fast network flooding. In other words, they do not rely on routed unicasts and are inherently robust to topology changes. In practice this means that if there is a path to a node, Chaos and related approaches will find it. Thus; this class of protocols is a strong alternative for robust routing protocols in challenging, dynamic environments. However, the design of Chaos shows limitations under [5]: (1) short-term interference during the end of round and (2) long-term interference.

If a node at the end of a round cannot receive valid packets due to interference, this node will most likely not reach completion and as a result stay awake until the round times out. This leads to (a) that node not having the final result of the computation, (b) consuming more energy. Other nodes, however, have received the result already and therefore will reach completion. In our experience, this critical time concerns the last 10 to 20 slots of a round. It is in the order of tens of milliseconds. If a node is under interference during a complete round, then neither this node nor the the rest of the network will reach completion, as the other nodes are waiting for input from this node. Overall, we argue that Chaos, due to its design, inherently provides robustness in the presence of low- and medium levels of interference, such as short bursts from microwave ovens and coexisting 802.15.4 traffic. However, it has limitations in case of stronger interference, such as coexistence with 802.11 traffic.

4 Making Chaos Dependable

Once channels become jammed for a long duration, i.e., more than a couple of milliseconds, we see reliability in Chaos dropping and both latency and energy consumption increasing. In the following, we discuss our extensions to build a robust Chaos under interference.

4.1 Channel Hopping

Inspired by Bluetooth and WirelessHART [7], we add channel hopping: Following a pseudo-random hopping-sequence, each time slot uses a specific channel. This increases robustness if a node is not reachable on one or more channels. Previously, we argued that whenever there exists a route to a node, Chaos would find it due to its flooding based mechanisms. With channel hopping, we can extend this observation: We argue that whenever there exists a route to a node on any channel, Chaos will find it. If there is no announced static 802.11 interference during the competition, we plan to use the full 16-channel spectrum. In case of strong 802.11 interference, we can exclude these from the hopping sequence, or leave it up to the adaptive black listing mechanisms explained later.

4.2 Multiple Channels in Parallel

In dense networks, we can further increase robustness and decrease time until completion by utilizing multiple channels in parallel. Thus, for each slot we now pick multiple channels and each node randomly picks one of these per slot. In practice we choose two to five concurrent channels, depending on network density. Using multiple channels in parallel increases both the resilience and the overall performance, as it allows Chaos to progress on multiple channels in parallel.

4.3 Adaptive, Local Blacklisting

Next, we extend our approach of using multiple channels in parallel with adaptive local blacklisting. Using RSSI measurements, we determine the noise level of a channel during a Chaos round. Based on the channel noise level and reception success rate, we then blacklist bad channels. We use this to improve the use of multiple channels in parallel: Without local blacklisting we would utilize each of the parallel channels with the same probability. With local blacklisting, we give a higher probability to good channels while limiting the use of bad ones. These decisions are made locally, as interference is commonly similar for neighboring nodes.

4.4 Global Blacklisting

To identify channels with intense interference, we use Chaos to reach an agreement on the best and worst channels in the network. Thus, next to using local blacklisting, we run additional rounds of Chaos to collect and vote on the channel quality similar to how we compute the maximum value within the network. This feature is helpful in networks with long-term static interference, such as heavy 802.11 traffic. It should be noted that a lightweight version of network-wide blacklisting can be performed alongside the main application without additional rounds. The cost is a couple of flags per packet, and a simple union of blacklists is enough.

5 Simple Data Collection with Chaos

Originally, Chaos was designed as all-to-all network communication primitive. For example, Chaos can collect data from all nodes in the network or compute aggregates such as a maximum sensor value within the network. For the competition setup, data shall only be collected from a single node and delivered over multiple hops to a sink node. Thus; Chaos round can stop when the root notices that the sink got the packet.

6 Limitations of Robust Chaos

Chaos and similar approaches such as Glossy and LWB are a designed for periodic operation: The system schedules its operation for a specific periodicity and all devices enter sleep modes in between. The competition, however, uses an event driven model: Each event shall be detected and communicated to the sink. Moreover, both the average and maximum rate of events are unknown. This setup is challenging for the periodic design of Chaos and we have to calibrate it to balance latency and energy consumption.

7 Conclusion and Future Work

In this paper, we present a robust version of Chaos under both short- and long-term interference. We extend Chaos with channel hopping, utilize multiple channels in parallel, and employ local and global blacklisting of channels. In addition, we adapt Chaos to the specific application and traffic requirements of the competition settings.

8 References

[1] S. Duquennoy, B. Al Nahas, O. Landsiedel, and T. Watteyne. Orchestra: Robust mesh networks through autonomously scheduled tsch. In *Proceedings of the 13th ACM Conference on Embedded Networked Sensor Systems*, pages 337–350. ACM, 2015.

[2] S. Duquennoy, O. Landsiedel, and T. Voigt. Let the Tree Bloom: Scalable Opportunistic Routing with ORPL. In *SenSys: Proc. of the ACM Conference on Embedded Networked Sensor Systems*, 2013.

[3] F. Ferrari, M. Zimmerling, L. Mottola, and L. Thiele. Low-power wireless bus. In *SenSys: Proc. of the ACM Conference on Embedded Network Sensor Systems*, 2012.

[4] F. Ferrari, M. Zimmerling, L. Thiele, and O. Saukh. Efficient network flooding and time synchronization with glossy. In *IPSN: Proc. of the ACM/IEEE International Conference on Information Processing in Sensor Networks*, 2011.

[5] O. Gerbert. Attack on the chaos sensor network protocol, 2015.

[6] O. Gnawali, R. Fonseca, K. Jamieson, D. Moss, and P. Levis. Collection tree protocol. In *SenSys: Proc. of the ACM Conference on Embedded Networked Sensor Systems*, 2009.

[7] HART Communication Foundation. HCF_SPEC-065, 2.4GHz DSSS O-QPSK Physical Layer Specification. *HCF_SPEC-065, Revision 1.0*, 2007.

[8] O. Landsiedel, F. Ferrari, and M. Zimmerling. Chaos: Versatile and efficient all-to-all data sharing and in-network processing at scale. In *SenSys: Proc. of the ACM Conference on Embedded Networked Sensor Systems*, 2013.

[9] O. Landsiedel, E. Ghadimi, S. Duquennoy, and M. Johansson. Low power, low delay: Opportunistic routing meets duty cycling. In *IPSN: Proc. of the ACM/IEEE International Conference on Information Processing in Sensor Networks*, 2012.

[10] K. Leentvaar and J. H. Flint. The capture effect in fm receivers. *IEEE Transactions on Communications*, 24(5):531–539, 1976.

[11] T. Winter, P. Thubert, A. Brandt, J. Hui, R. Kelsey, P. Levis, K. Pister, R. Struik, J. Vasseur, and R. Alexander. RPL: IPv6 Routing Protocol for Low power and Lossy Networks. RFC 6550, IETF, 2012.