Interference Scripting: Protocol-aware Interference Generation for Repeatable Wireless Testbed Experiments

Matthias Wilhelm
Computer Science Department
TU Kaiserslautern
67653 Kaiserslautern, Germany
wilhelm@cs.uni-kl.de

Jens B. Schmitt
Computer Science Department
TU Kaiserslautern
67653 Kaiserslautern, Germany
jschmitt@cs.uni-kl.de

ABSTRACT

Despite its practical importance, interference can still be considered a “black box” in wireless network experiments as it is difficult to generate in a controlled and repeatable manner. Current generation approaches, such as packet storms or pre-recorded interference traces, do not adapt to the transmissions on the channel; the resulting effects of the interference are random and beyond the experimenter’s control. Our solution to this problem is to use channel-aware interferers, allowing them to adapt to the actual packet transmissions on the channel. We implemented a reactive jamming system on the USRP2 that enables this mode of operation as a proof of concept, decoding packets and interfering with them during their transmission. With this capability in mind we propose *interference scripting*, a way to define protocol-aware interference patterns using both packet content and time, and to repeatably generate these patterns on dedicated devices deployed alongside a testbed. This way, we hope to provide a useful tool to experimenters, adding controllable interference to wireless testbeds.

Categories and Subject Descriptors

C.4 [**Performance of Systems**]: *Measurement techniques*; C.2.1 [**Computer Communication Networks**]: Network Architecture and Design—*Wireless communication*

General Terms

Design, Experimentation, Measurement

Keywords

Interference, USRP2, RFReact, SDR, Sora, testbeds, WLAN, WSN

1. INTRODUCTION

Shared spectrum is a widely used concept because it makes it easy to develop and deploy wireless networks, as long as the devices do not obstruct or frequently interrupt other services in the band. Because of this requirement, networks use collision avoidance mechanisms such as carrier sense and random backoff to ensure fair access to the medium. However, even today the spectrum is already crowded by an increasing number of wireless devices (using a plethora of standards such as WLAN, ZigBee, Bluetooth), posing a challenge to network protocol designers and operators alike. The resulting interference is a major factor in the performance and reliability of wireless networks: it adversely affects key metrics such as throughput, packet loss, and delay.

The performance of a wireless network under interference is hard to predict for several reasons. First, a packet may still survive interference depending on the circumstances: the positions of sender, receiver, and interferer; the propagation environment; the resistance of the receiver against the particular type of interference; or the use of coding to recover damaged packets (e.g., [4]). Second, the interference may exhibit patterns that harm network performance disproportionately, e.g., by destroying a packet and all corresponding retransmissions. Analytical treatment or simulations of network protocols often require simplifying assumptions to keep the evaluation treatable. For example, instead of complex channel models that are required to capture physical wave interactions in realistic environments, only simple fading models such as path distance or log-normal shadowing are often used. Another example is that the concurrent use of different medium access protocols may break the assumption of independence of interference events, which is often assumed to keep the analysis simple.

Because of these issues real-world experiments are widely used to explore the performance of protocols in realistic settings. The goal is to show that the protocol performs as intended in a representative environment. Still it is very challenging to reproduce experimental results in other environments (and oftentimes even in the same environment). To increase comparability and repeatability, researchers started to build and share testbeds where experiments can be performed under stable conditions. Yet, even in these controlled environments, interference is still problematic in experimentation, especially its repeatable and reliable generation. A common approach to interference generation is to deploy COTS devices in the testbed (e.g., using WLAN access points or sensor motes) and program them to send packets at random times or with a fixed rate. While this approach yields insights in the performance of a protocol in crowded settings, it is neither realistic nor repeatable [1]. Another approach is to record interference patterns and replay them during the experiment. For example, Boano et al. [2] show that sensor motes may be programmed to precisely capture and generate interference in a sensor network testbed.

Despite these efforts, the effects of interference are still a “black box” to experimenters. Even if exactly the same interference pattern is reproduced, the response of the net-

S3’12, August 26, 2012, Istanbul, Turkey.
ACM 978-1-4503-1528-9/12/08.

work may deteriorate from its previous behavior because of small timing differences or internal system effects, leading to differing execution traces and a different performance. Oftentimes the response of the network is the parameter of interest (e.g., the packet reception ratio), not the duty cycle of the interferers. What we need is protocol-aware interference that also takes the state of the network into consideration, allowing direct control of these parameters. We propose a system that addresses this need and enables both repeatable and more realistic experiments.

In previous work [6] we presented a reactive jamming system that supports precise interference generation and channel-aware operation. In this extended abstract, we describe our on-going work to extend this system to support "interference scripting," a way to specify portable interference patterns that are protocol-aware, i.e., adapt the interference according to the network's operation. The goal is to allow a full offline specification of the complete network interference, and repeatable and real-time execution during the experiment.

2. INTERFERENCE SCRIPTING

The idea is to deploy dedicated interferers in the testbed that are programmed and controlled individually from a central point. Each device monitors the channel, detects packets, and generates interference in response to packet and timing events according to a supplied script. We briefly describe the implementation of the interference generation nodes and their operation, and the necessary changes to support scripting for a wide range of application scenarios.

2.1 The Dedicated Interferer System

We built a channel-aware jamming system (RFReact [7]) that allows to access the content of an IEEE 802.15.4 packet and still interfere with the end of the packet. It is implemented on the USRP2, an affordable software-defined radio platform. No modifications to the hardware are necessary, RFReact is implemented in software only. Still, because we use the USRP2's FPGA, it achieves reaction times in the order of tens of microseconds, allowing to listen to the largest part of the packet and still destroy it. This mode of operation is also reliable: our experiments show that 99.9 % of the targeted packets are actually destroyed while other packets are correctly classified and remain unaffected.

We note that the concept of interference scripting is not limited to IEEE 802.15.4 networks. We are also experimenting with the Sora platform [5] to bring this concept to IEEE 802.11 b/g networks.

2.2 Scripting

In previous work, we used this system to implement a classification system similar to firewall rules [8]. While this concept is already sufficient for some realistic interference scenarios, there are more components required for a fully featured experimental system. The vision is that these components can be combined to define arbitrary interference patterns in a script, using control structures like loops and conditions. The script ties together the following conditions to select packets for interference:

Packet content: Access to the packet content is an important step to protocol-awareness; it allows to restrict interference to packets with a chosen source/destination, to packet types, or bits in the payload. The real-time demodulation of RFReact allows this access to the content $4\,\mu$s after the corresponding physical layer symbol was on the air.

Timing: To support pre-recorded interference patterns or to occupy the channel for measuring the performance of carrier-sense based protocols, the system must also support scheduled interference. The USRP2 supports a timing precision of 10 ns and transmission timestamps that allow this mode of operation.

Randomness: Because blocking all packets of a kind is not a realistic interference scenario (when no adversarial setting is considered), we need a way to define interference patterns that follow random distributions of choice, e.g., 30 % of the packets with bursty interference. This feature requires the implementation of random sources with distributions of interest.

Protocol state: State information enables interference decisions with memory. Taking a trace of packets into consideration, this enables *white box* testing of wireless protocols, triggering interference only when a sequence of packets was previously detected. For example, a node may loose its connection to surrounding nodes each time it successfully associates with the network.

Interference waveforms: To generate interference patterns using multiple communication standards, or to mimic unintentional interferers such as microwave ovens, we need a way to choose interfering waveforms on a per-packet basis. The USRP2 supports arbitrary sequences of samples as digital representation of waveforms, allowing to store or compute them on-the-fly.

3. APPLICATION EXAMPLES

Next, we describe some possible applications of interference scripting.

Protocol-aware interference: With our approach, it is possible to define fine-grained interference patterns, e.g., targeting specific MAC layer packets such as ACKs to debug the interaction between application and OS layer software (this approach was used in related work to discover race conditions in the Contiki operating system [3]). However, in contrast to that work, our system does not rely on timing information, such that we can decide if interference is required and start it before the packet is over. An application of this is to target neighborhood discovery messages (used in routing protocols) to observe the behavior of the network in such adverse conditions.

Virtual topologies: By selectively interfering with packets based on their header addresses, we can define and enforce virtual topologies by blocking a subset of neighbors, or making links between nodes directional. This increases the control of the experimenter over the topology, and adds flexibility to an existing testbed. For example, it can alleviate the need of physically rearranging devices in the network, or help to evaluate the behavior of protocols in changing environments rapidly.

Arbitrary loss processes: Targeting packets directly allows to choose which random distribution the packet loss process should follow. This allows to compare the performance of protocols under the same conditions, e.g., using 70 % packet loss with interference bursts. Even if the protocols are operating differently, e.g., using different medium access strategies, we can control the interference patterns precisely because we are not relying on timing but targeting packets directly. This is also important when comparing results to simulations because the same random distributions can be used. This enables direct comparison between simulated results and observations from the real system.

Arbitrary waveforms: Emulating different devices in the vicinity (microwave ovens, baby phones, WLAN or ZigBee devices) is also an important application, which can be combined with the previous strategies to choose the interfering shape for each packet individually.

4. CONCLUSION

We outlined the operation and the possible uses of our approach to generate protocol-aware interference, to make the interference controllable by experimenters and to increase the repeatability of experiments by "interference scripting." dWe plan to release our system as open source code to support researchers with their experiments.

5. REFERENCES

[1] C. A. Boano, Z. He, Y. Li, T. Voigt, M. Zuniga, and A. Willig. Controllable radio interference for experimental and testing purposes in wireless sensor networks. In *Proc. of IEEE LCN '09*, pages 865–872, Oct. 2009.

[2] C. A. Boano, T. Voigt, C. Noda, K. Römer, and M. Zuniga. JamLab: Augmenting sensornet testbeds with realistic and controlled interference generation. In *Proc. of IPSN '11*, pages 175–186, 2011.

[3] Z. He and T. Voigt. Precise packet loss pattern generation by intentional interference. In *Proc. of IEEE DCOSS '11*, pages 1–6, June 2011.

[4] C.-J. M. Liang, N. B. Priyantha, J. Liu, and A. Terzis. Surviving wi-fi interference in low power ZigBee networks. In *Proc. of ACM SenSys '10*, SenSys '10, pages 309–322, 2010.

[5] K. Tan, J. Zhang, J. Fang, H. Liu, Y. Ye, S. Wang, Y. Zhang, H. Wu, W. Wang, and G. M. Voelker. Sora: high performance software radio using general purpose multi-core processors. In *Proc. of USENIX NSDI '09*, pages 75–90, Apr. 2009.

[6] M. Wilhelm, I. Martinovic, J. B. Schmitt, and V. Lenders. Reactive jamming in wireless networks: How realistic is the threat? In *Proc. of ACM WiSec '11*, pages 47–52, June 2011.

[7] M. Wilhelm, I. Martinovic, J. B. Schmitt, and V. Lenders. RFReact—a real-time capable and channel-aware jamming platform. *SIGMOBILE MCCR*, 15:41–42, Nov. 2011.

[8] M. Wilhelm, I. Martinovic, J. B. Schmitt, and V. Lenders. WiFire: A firewall for wireless networks. In *Proc. of ACM SIGCOMM '11*, pages 456–457, Aug. 2011.