SWAT: Enabling Wireless Network Measurements

Kannan Srinivasan, Maria A. Kazandjieva, Mayank Jain, Eddie Kim, and Philip Levis
Computer Systems Lab
Stanford University
Stanford, CA 94305
{srikank,mariakaz,mayjain,edskim}@stanford.edu, pal@cs.stanford.edu.

Abstract

Measuring low-level wireless network properties allows researchers to understand how protocols and applications perform in different environments. In this demo, we present SWAT – a software tool that automates gathering and analysis of such network measurements. SWAT provides an interface for configuring experimental parameters in a network. It collects raw packet statistics such as the received signal strength, chip error, and sequence number, and provides modules for calculating and visualizing various metrics derived from these statistics.

Categories and Subject Descriptors: C.4 [Performance of Systems]:Measurement techniques C.2.1 [Network Architecture and Design]:Wireless communications

General Terms: Experimentation, Measurement, Performance.

Keywords: 802.15.4, 802.11, Wireless measurements, Network Metrics.

1 Introduction

Measuring wireless network performance in terms of high-level metrics such as end-to-end throughput, delivery ratio, and latency is common practice. Yet, when deploying a protocol, it is difficult to understand exactly what aspects of the network were responsible for the observed performance. Therefore, it is increasingly important to report lower level metrics such as temporal variations in packet delivery [2, 7] and correlation of network attributes over space [4]. Knowledge of the underlying dynamics of the environment can give insight into why a protocol worked as expected or did not work at all, as well as guide future protocol designs.

Researchers have identified several low-level metrics that enhance the understanding of network dynamics [8, 3, 6, 5, 7]. However, there is no unified tool that runs experiments, gathers data and reports these metrics for an arbitrary network. The Stanford Wireless Analysis Tool (SWAT) provides these capabilities to the wireless community.

2 SWAT

SWAT allows researchers to easily gather network data, distill that data into relevant metrics, and visually display the results.

2.1 Description

Figure 1 presents the main structure of the SWAT software tool. Each component has a dedicated purpose and the interfaces between components give the flexibility of swapping out or adding new modules.

Users specify the experimental parameters through a configuration UI. These parameters define the settings for which packet and link data will be collected. They include link-layer type (802.15.4/802.11), node list, number of packets, inter-packet interval, type of transmission i.e. broadcast or unicast, CSMA on/off, channel, transmission power, link layer acknowledgements on/off, link layer retransmissions on/off (for unicasts), maximum retransmission count, bit rate (for 802.11), noise sampling on/off and noise sampling rate.

SWAT uses the appropriate interface to program the nodes, send commands and receive packet statistics through a wired or wireless back-channel. SWAT stores the collected data into a database using an SQL server. The reporting UI component allows users to specify which of the supported metrics they would like to calculate from the stored data. SWAT creates reports that consist of the experimental parameters, computed metrics and pertinent figures.

Currently, supported metrics include packet delivery temporal and spatial correlations, noise floor distribution, received signal strength to reception ratio correlation, link asymmetries, and reception ratio distribution over time.

2.2 Implementation

SWAT is an open-source, platform-independent tool. The user interface portions are written in Java and the communication interface between SWAT and the wireless sensor network uses python. Platform-specific code

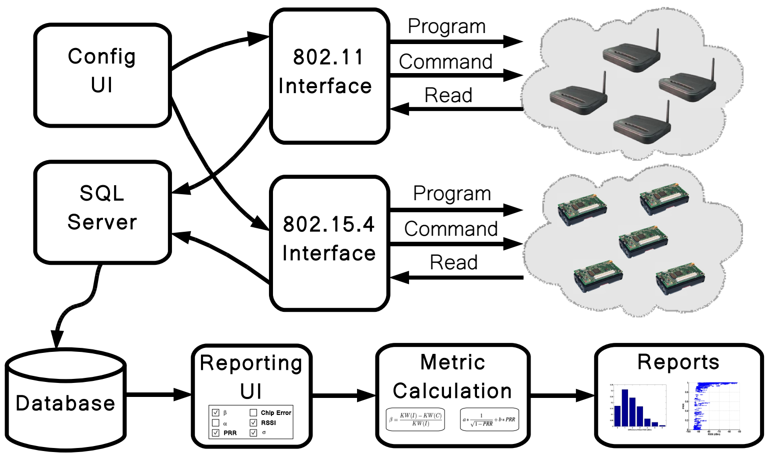

Figure 1. SWAT provides components for programming wireless nodes, collecting and storing raw packet data, computing network metrics, and generating reports. The modular structure allows users to extend its capabilities.

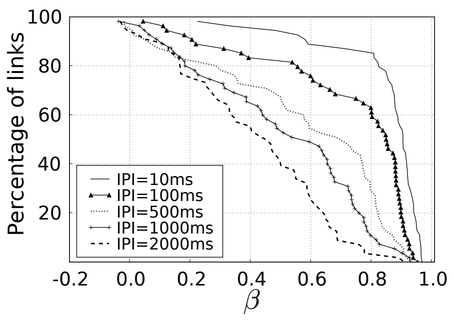

Figure 2. Complimentary cumulative distribution of β. The experiment consisted of 100,000 broadcast transmissions from each node with different inter-packet intervals (IPI) on channel 26.

that runs on the sensor motes is written in nesC, an extension to the C programming language for TinyOS. In the future, SWAT will also support 802.11 networks.

We use SQL to create the database, making it easy to execute custom queries. The computation intensive component of SWAT is the metric calculation. It consists of a number of python programs, each responsible for computing one metric. SWAT is extensible so users can write their own python modules.

3 Discussion

SWAT provides a framework for measuring and reporting network properties. This allows researchers to comprehend protocol performance in a specific network context and also enables meaningful comparison of protocol performance across environments.

For example, Figure 2 presents the complimentary cumulative distribution of β [7], a temporal correlation measure, generated using SWAT for an experiment run on an 802.15.4 testbed. Most of the links on the testbed have temporally correlated delivery. In prior work, we used this information to increase the efficiency of Collection Tree Protocol (CTP) [7]. This example illustrates how SWAT metrics can lead to protocol improvements. Knowledge of low-level network characteristics can also help understand the gap between testbed and real-world performance discrepancies. In addition, SWAT measurements will ensure a common basis for comparing protocols.

SWAT will allow researchers to identify new metrics that may be relevant to specific protocols and to easily share low level measurements with the research community using public databases such as CRAWDAD [1]. In the future, we envision that SWAT metrics measured empirically can be used as simulation parameters to enable test repeatability.

In the demonstration, we will show how SWAT can be used to configure measurement experiments in a 4-node ad-hoc sensor network. In addition, to show the full capabilities of the metric calculation and reporting components, we will use a larger dataset collected prior to the demo.

4 References

[1] A Community Resource for Archiving Wireless Data At Dartmouth. `http://crawdad.cs.dartmouth.edu`.

[2] D. Aguayo, J. C. Bicket, S. Biswas, G. Judd, and R. Morris. Link-level measurements from an 802.11b mesh network. In *SIGCOMM*, pages 121–132, 2004.

[3] A. Cerpa, J. L. Wong, M. Potkonjak, and D. Estrin. Temporal properties of low power wireless links: modeling and implications on multi-hop routing. In *MobiHoc '05: Proceedings of the 6th ACM international symposium on Mobile ad hoc networking and computing*, pages 414–425, New York, NY, USA, 2005. ACM Press.

[4] N. Patwari and P. Agrawal. Effects of correlated shadowing: Connectivity, localization, and rf tomography. In *IPSN '08: Proceedings of the 2008 International Conference on Information Processing in Sensor Networks (ipsn 2008)*, pages 82–93, Washington, DC, USA, 2008. IEEE Computer Society.

[5] D. Son, B. Krishnamachari, and J. Heidemann. Experimental analysis of concurrent packet transmissions in low-power wireless networks. Technical Report ISI-TR-2005-609, Nov. 2005.

[6] D. Son, B. Krishnamachari, and J. Heidemann. Experimental analysis of concurrent packet transmissions in low-power wireless networks. In *SIG-COMM*, 2006.

[7] K. Srinivasan, M. A. Kazandjieva, S. Agarwal, and P. A. Levis. The β-factor: Measuring wireless link burstiness. In *SenSys '08: Proceedings of the 6th international conference on Embedded networked sensor systems*, Raleigh, NC, USA, 2008. ACM Press.

[8] J. Zhao and R. Govindan. Understanding packet delivery performance in dense wireless sensor networks. In *Proceedings of the First International Conference on Embedded Network Sensor Systems*, 2003.