CENS Center for Embedded Networked Sensing

Non-Intrusive Analysis of Sensor Network MAC Protocols

Tyler McHenry, John Heidemann

ISI Laboratory for Embedded Networked Sensing Experimentation - http://www.isi.edu/ilense

Introduction: Some experiments require MAC-layer analysis of live sensor networks

Example: High-Density Sensor Networks

- **High density networks cannot be simulated reliably**
  - As potential traffic increases, the actual collision rate must be measured rather than modeled statistically to discover realistic MAC performance In real networks, packets do not collide or corrupt at a known rate.
  - Low-level MAC layer ***timings*** need testing in light of real collision rates.

Needs/Requirements:

- **Software to collect and analyze the behavior of active sensor networks**
  - Must not interfere with the operation of the network
  - Must be able to deliver state information which is not explicitly communicated between nodes

Problem Description: MAC-Level analysis is hindered by the addition of reporting code

- **Extra traffic generated**
  - Sending reporting data over the network requires either piggybacking it on normal packets or creating entirely new packets for debugging purposes, increasing traffic on the network.
  - Plus, the delivery of this data depends on the ***reliability*** of the network, which may be a variable!
- **Tethered nodes**
  - Sending reporting data over backchannels restricts the placement of nodes and cannot be used in an existing deployment.
- **Important MAC timings affected**
  - In S-MAC, the extra processing and transmission time necessary to add extra debugging information to SYNC packets causes the S-MAC period to elongate by 1.04 ms, which may affect performance under dense or heavy-use conditions.
- **Inability to detect some collisions**
  - Relying on the source to report its transmissions ignores collisions entirely.
  - Relying on the sink to report its receptions will still fail to capture ***"total loss" collisions*** (in which preambles are corrupted), without modifying the MAC.

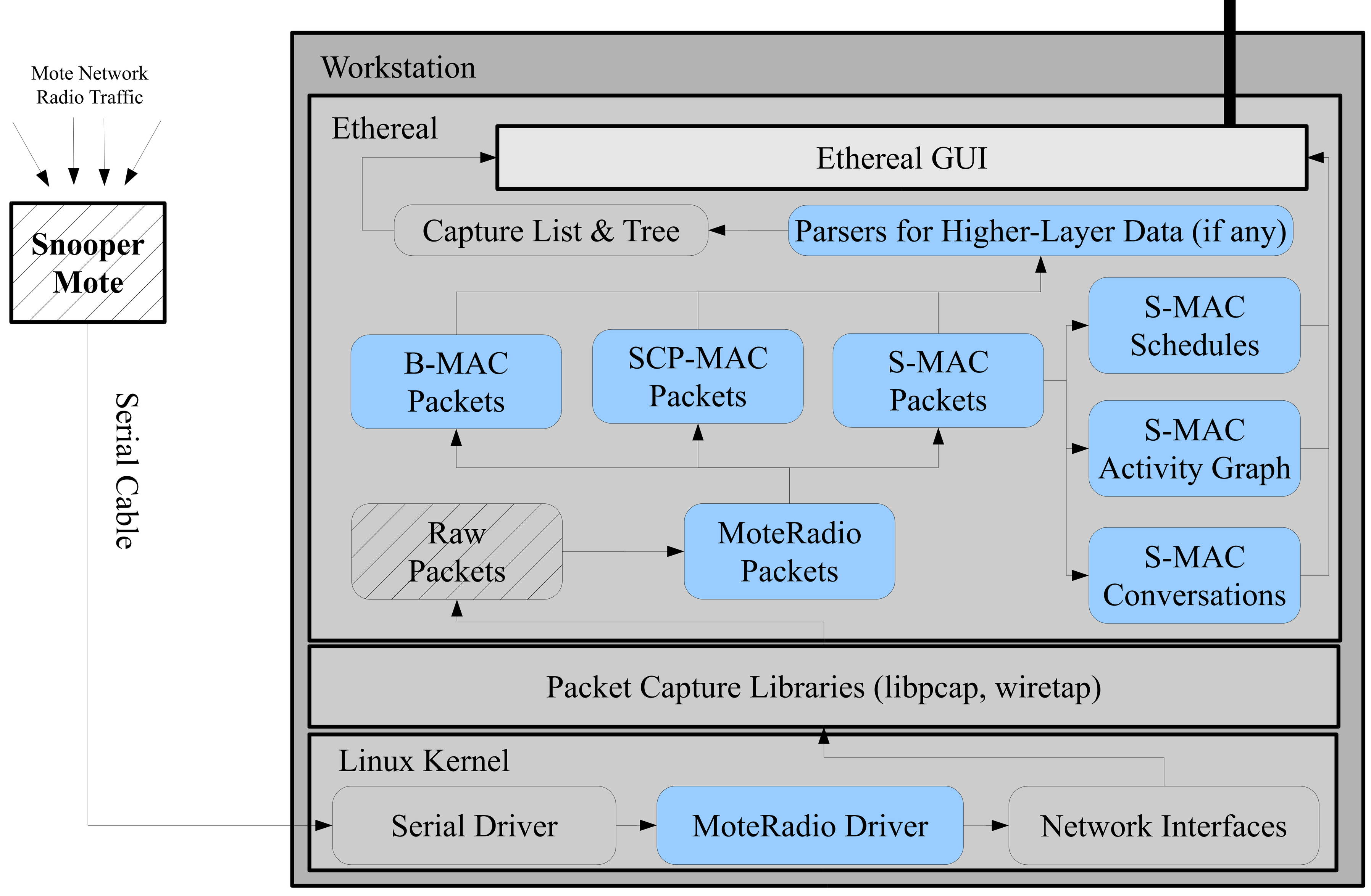
Key Idea:

Completely avoid modifying all mote-side software

Proposed Solution: A snooper mote can collect data to be processed later with Ethereal addons

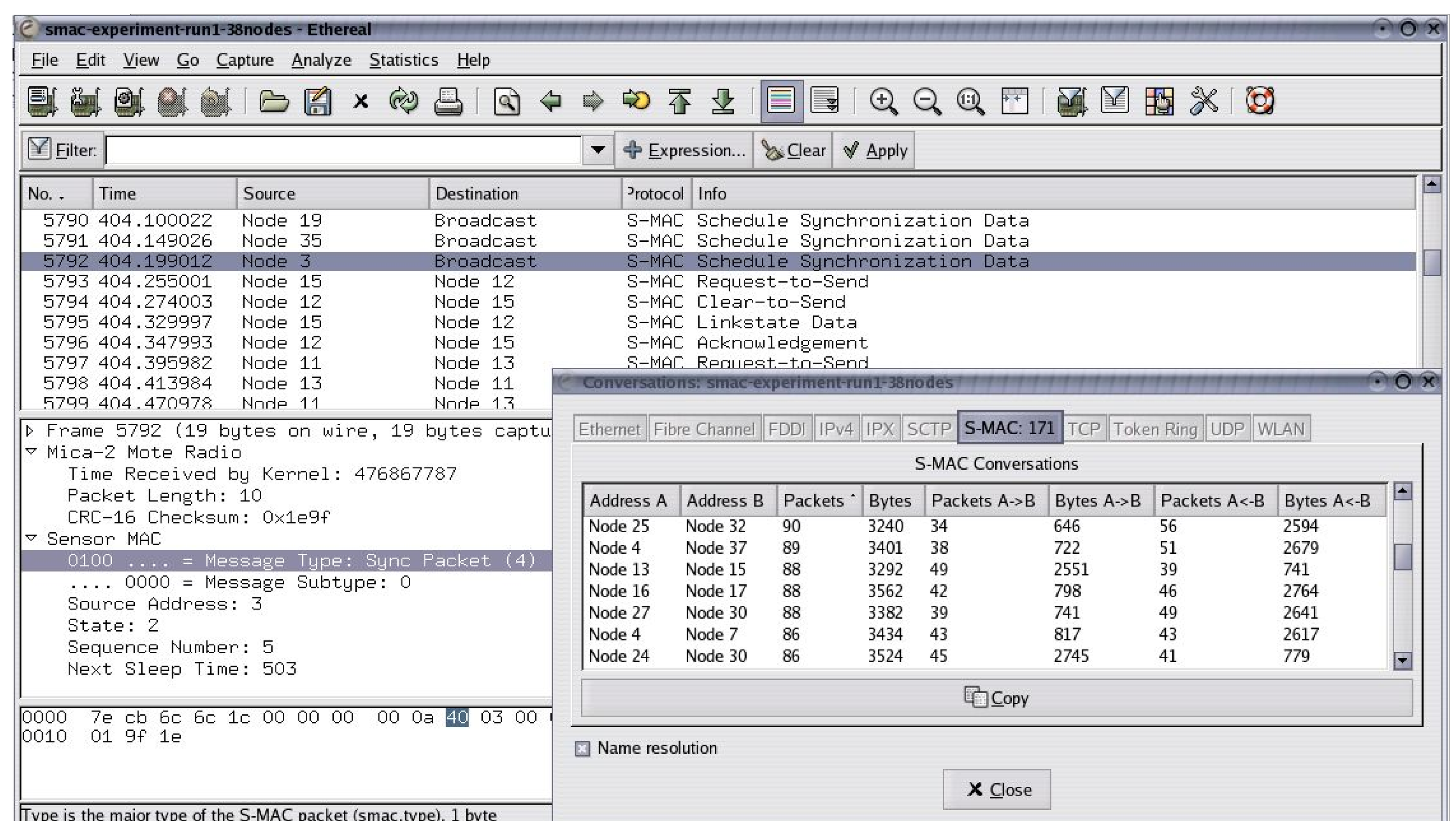
The "Radio Traffic Analysis" (RTA) Suite

- **Data collection based on *snooper motes***
  - Constantly listens using the same physical layer as the MAC being analyzed, but need not implement the MAC protocol itself.
  - Echoes every byte it overhears to a Linux host over some backchannel (most simply, a serial cable). Other motes do not even know it exists.
  - Can also monitor for "total loss" packet collisions by detecting changes in RSSI.
- **Packet-level processing emulates a Linux network interface**
  - The ***moteradio driver*** accepts bytes from a snooper mote and presents them as whole packets on a network interface with a microsecond timestamp added.
  - Any program can read this interface as it would read any other network card.
- **Reporting and analysis software are addons to Ethereal**
  - A familiar and widely-used network traffic analysis package
  - Provides a portable framework and familiar GUI that can capture packets from any network interface and feed them to custom processing code depending on the type of packet captured.
- **New *dissectors* (packet format definitions) are easily created for new MAC protocols or higher-level protocols**
  - Packets progress through a tree of analysis code from physical layer to MAC layer to higher layers (if applicable).
  - Adding a new dissector for a new MAC or higher layer protocol is straightforward.
  - Dissectors currently exist for B-MAC, S-MAC, SCP-MAC, the SMACTest application, and Linkstate data.
- **Designed from the bottom up to be as portable as possible**
  - Ethereal is already portable to Linux, Windows and more.
  - Porting to a different version of Linux or a new platform entirely necessitates modifying or replacing the moteradio driver only.



Blue elements are completely new, hashed elements are modified.

The Ethereal GUI with RTA Modifications



Screenshot of Ethereal displaying an S-MAC SYNC packet and a list of point-to-point conversations determined from this flow of packets

Goal: Automatic Adaptability in MAC Analysis

- **Each protocol with a dissector may provide more complex data analysis**
  - For S-MAC, RTA provides: point-to-point conversation tracking, enumeration of schedules and schedule tracking, and node activity graphs.
  - Each of these requires intensive post-processing of captured packets.
- **Analysis provided by the dissector should not make assumptions**
  - A key use of Radio Traffic Analysis is testing the effectiveness of modifications to protocol parameters.
  - Dissectors should *detect* parameters whenever possible.
- **Example: S-MAC sleep/listen cycle length (period) and schedules**
  - The S-MAC dissector *detects* the period in use within some margin of error and is able to determine that 3 nodes are on the same schedule
  - It can then constrain the timing of sleeps in that schedule within a +/- 5ms window for prediction of future sleep times.