Capturing *Bluetooth* Traffic, the Right Way

Introduction

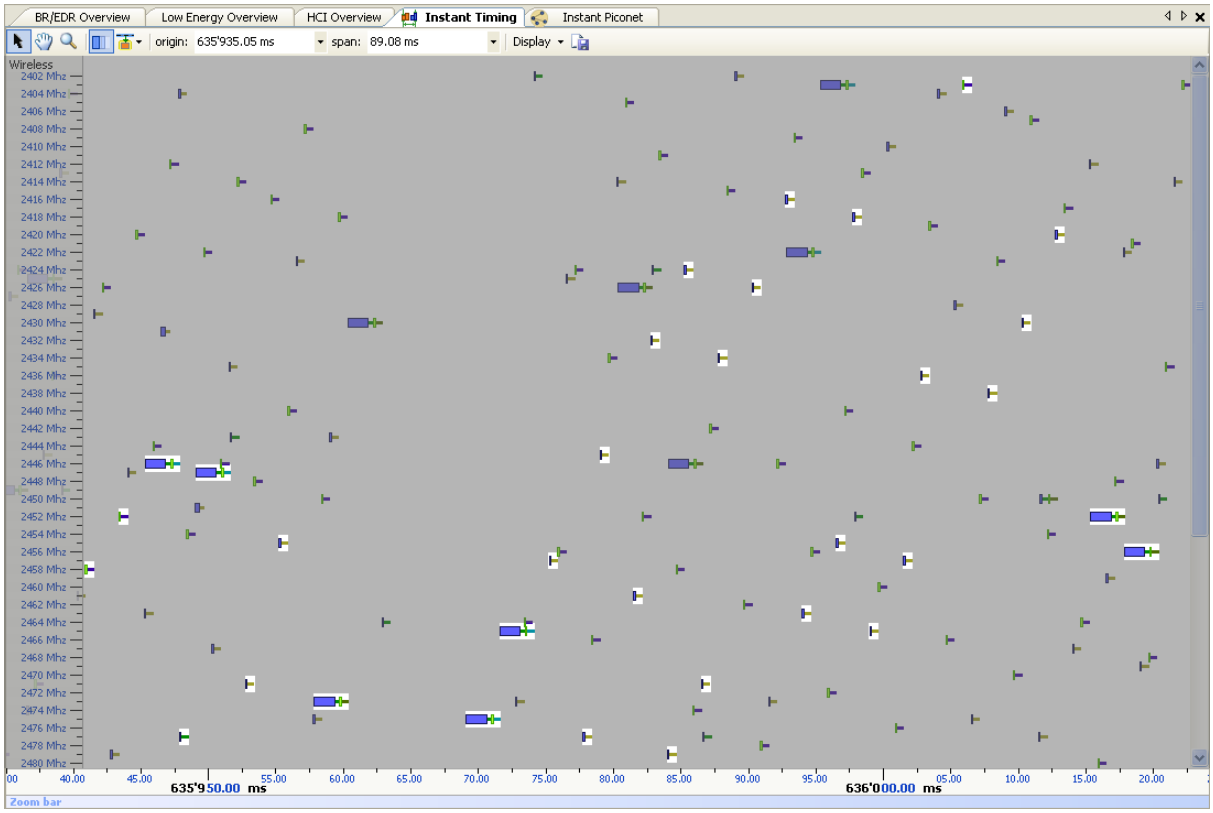
Bluetooth is a difficult technology to sniff. Advanced communication techniques such as frequency hopping, whitening, and encryption are all characteristics that make this ever-evolving technology "sniffer unfriendly".

This paper will cover how these challenges affect engineers and what we can do to improve the situation. We will specifically cover wireless capture, focusing on the two sniffing techniques available today: single- channel sniffing and whole-band concurrent channel sniffing.

Single-Channel Sniffing

Single-channel sniffing is the most common sniffing method, as it is used by all sniffers that have been introduced on the market from *Bluetooth*'s inception until recently. This sniffing method involves listening to just one of the 79 available channels at a given time. As *Bluetooth* devices use a different channels every 625us, this sniffing method needs to know very precisely when and where to listen in order to capture the packets from a given piconet.

The following illustration shows *Bluetooth* packets from several piconets. The 79 *Bluetooth* channels are represented vertically, with time progressing horizontally. The highlighted packets are captured by the single-channel sniffer, while all other packets from other piconets are not seen.



This sniffing method can be implemented with a standard radio chip and specific firmware. A simple, single-channel sniffer, more or less acts similarly to a standard device. It follows the hopping sequence of the selected piconet, and

captures the packets in each slot. But before being able to capture any packet, it will need to synchronize to the piconet of interest. The simple sniffer manages this synchronization just as any device will: it pages the master of the piconet by sending ID packets. When the master sees these packets, it transmits an FHS. This FHS contains all information required in order to sync on the master and to follow the hopping sequence. From this point, and only from this point, the simple sniffer can receive packets from this piconet.

So far so good, but this approach has a few limitations.

The first and most obvious is that this approach **cannot capture unsynchronized traffic**, such as inquiries or pagings. Moreover, when the sniffing process doesn't succeed the user has no clue why this has happened. Was the slave in an unstable state, or was the master? Or maybe was the simple sniffer incorrectly configured?

Another issue relates to maintaining synchronization. Since the simple sniffer is following the master, and since the master is not aware of the sniffer, the master will not care about it. For example, when the master has no active slave, it does not transmit any traffic, and the simple sniffer will not be able to stay in sync with it for a long period of time, because of clock drift. The simple sniffer will then need to do a paging again, resulting in **several seconds of blindness**, which can be very frustrating. Sometimes you get a capture, sometimes not.

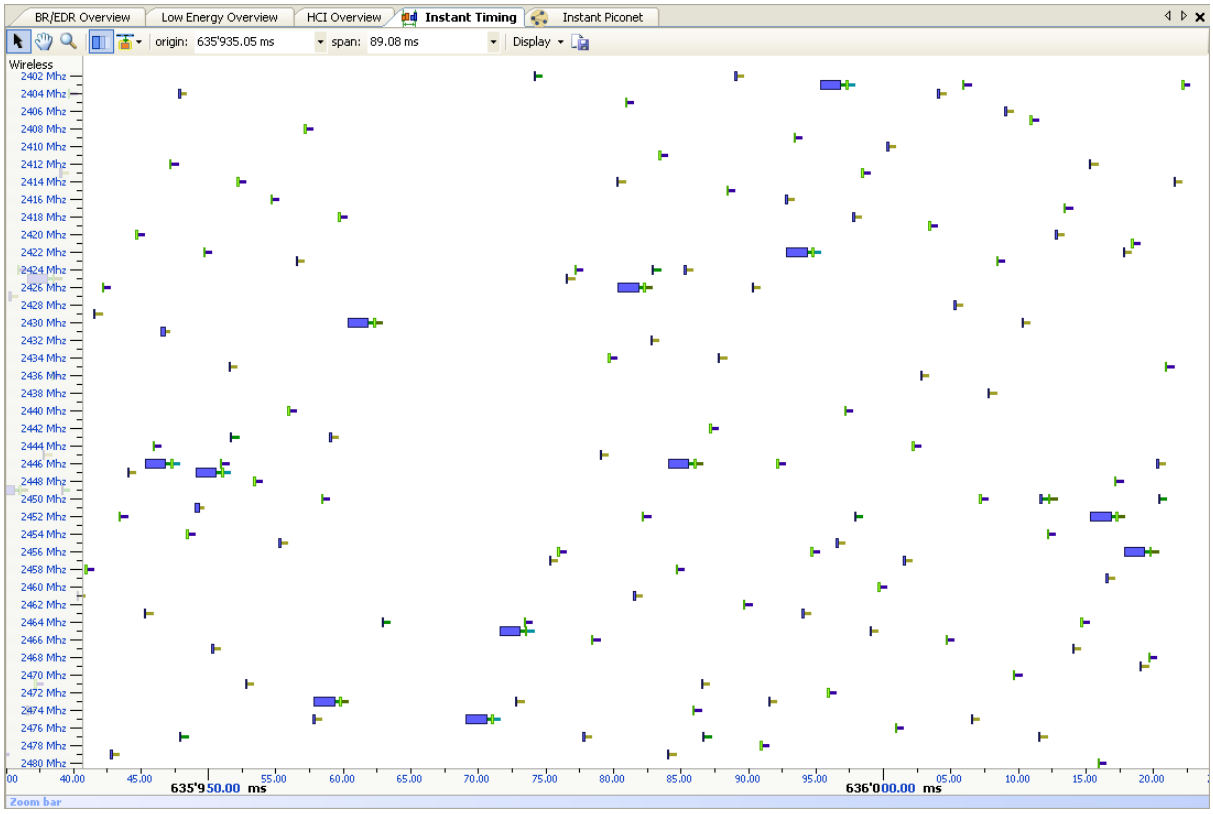
Another limitation (and maybe the most critical) relates to encryption. The good old days of basic PIN-code based security are over; new devices are all using Secure Simple Pairing (SSP). This makes debugging trickier, especially with off-the-shelf devices. The primary issue with a single-channel sniffer is that the sniffer **must decrypt the traffic in real-time in order to correctly follow the piconet**. This is for example required in order to follow adaptive frequency hopping (AFH), which is configured using encrypted LMP requests. Because of this limitation, single-channel sniffers are not capable of capturing the traffic in the first place, and then later decrypting the traffic by post-processing. This limitation also forbids simple sniffers to capture an SSP pairing followed by traffic of interest for example, which is quite limiting.

Last but not least, a single-channel sniffer can only follow a **single piconet**, which is less and less acceptable with modern use cases, which are increasingly requiring more complex topologies.

Whole-Band Concurrent Channel Sniffing

A more interesting way to sniff *Bluetooth* is to listen to all 79 *Bluetooth* channels concurrently. Instead of trying to follow a hopping sequence, such a sniffer will **listen to all channels**, and as soon as any packet is transmitted on any channel, it will be captured. This way, the sniffer hardware doesn't have to worry anymore about piconets, it can just **capture any traffic, statelessly, without any configuration**. All intelligence compilations are then done by post-processing in software.

The image below represents packets from several piconets, which have been captured by the Ellisys whole-band sniffer.



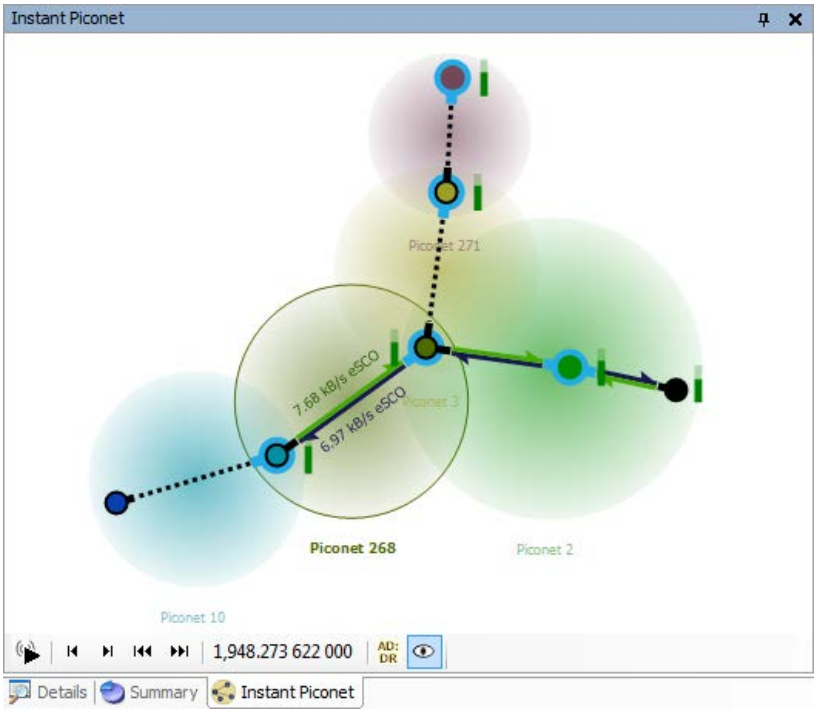
The Ellisys *Bluetooth* Explorer 400 is the first and only sniffer to employ a whole-band sniffing technique. Such sophistication cannot be achieved with standard *Bluetooth* radio chips. Ellisys engineers designed from scratch a **proprietary whole-band radio**, including an equipment-grade RF front-end, concurrent-channel digital radio and baseband.

Importantly, this approach makes the Ellisys BEX400 particularly **future-proof**. New paging techniques, baseband improvements, security enhancements, and added protocols & profiles can be accommodated with software updates alone, without changing anything in the hardware.

Another great advantage of whole-band sniffing is that capturing **asynchronous traffic** is no longer an issue. Paging and inquiries can be captured flawlessly. This capture method is intrinsically insensitive to role switches, adaptive frequency hopping, paging schemes, etc. Capturing marginal traffic is also not an issue: even if a packet is transmitted before, at the limit, or after the time slot, it will still be captured. And with its sub-symbol 125ns timing precision, the BEX400 provides unmatched fidelity. Its super-precise, temperature-stabilized internal oscillator makes it the perfect reference when measurements have to be made in between various devices.

This approach also opens new possibilities when capturing encrypted connections. As the Ellisys sniffer does not need to interpret *Bluetooth* traffic, it can capture encrypted traffic and decrypt it afterwards, in post-processing. This allows for automatic determination of the PIN-code, capture of SSP Debug Mode devices, and capture of SSP pairings, followed by decryption of 100% of the traffic by simply providing the link key afterwards, which is not possible with other sniffing techniques.

Last but not least, this sniffer method enables capture of not just a single piconet, but **all piconets and scatternets in the neighborhood**. Debugging complex topologies is becoming more and more important as use cases are quickly evolving from simple point-to-point connections to multi-profile, multipoint-to-multipoint connections.



Feedback

Feedback on our Expert Notes is always appreciated. To provide comments or critiques of any kind on this paper, please feel free to contact us at expert@ellisys.com.

Other interesting readings

- EEN_BT03 - Your First Wide-Band Capture
- EEN_BT06 - Bluetooth Security - Truths and Fictions
- EEN_BT07 - Secure Simple Pairing Explained
- More Ellisys Expert Notes available at: http://www.ellisys.com/technology/expert_notes.php

Rev. A. Updated 2011-05-16