University of BRISTOL

FACULTY OF ENGINEERING
DEPARTMENT OF ELECTRICAL & ELECTRONIC ENGINEERING

Networked Systems and Applications
EENGM0009

IEEE 802.15.4 / 6LoWPAN / RPL
Network Analysis with Wireshark

Dr George Oikonomou
March 2021

1 Introduction

This document aims to guide you through the analysis of an IEEE 802.15.4 / 6LoWPAN / RPL network with Wireshark[1].

The instructions here are with reference to a packet capture (pcap) file that was collected from a simulated 6LoWPAN. The pcap is available through blackboard to students enrolled on the Networked Systems and Applications unit (EENGM0009).

Question:

- *You will see various parts of the document starting with the word "Question" and formatted like this.*
- *These are prompts for you to think and dig around the packet capture.*
- *You are not assessed on those and you do not need to send me any answers!*

2 Network Topology and Device Operation

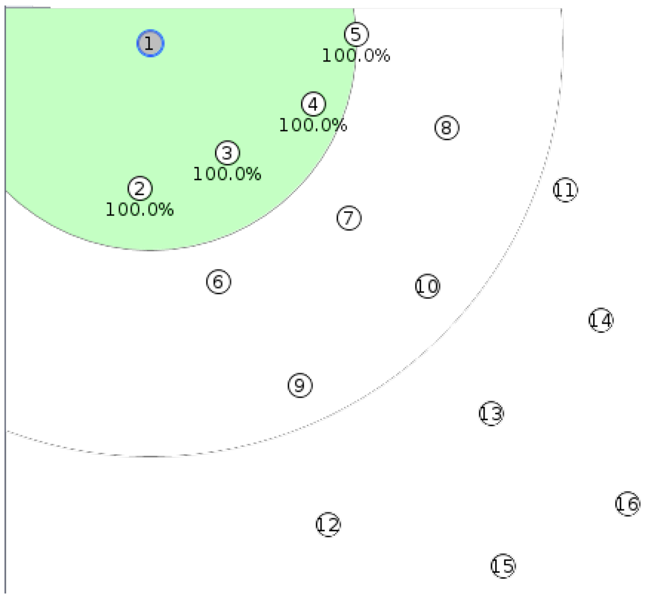

Figure 1: Simulated Network Topology

Figure 1 shows the topology of the simulated network:

- Node number 1 is the RPL root.
- The green circle represents node 1's radio transmission range. Every device within that green area can communicate with the root directly at the link layer.
- The second, outer circle represents the root's interference range. Frames transmitted by the Root will cause interference to devices within that area, but these devices will not be able to correctly receive those frames.
- Assume similar transmission/interference ranges for all other devices.
- RPL operates in MOP2.
- Each of the other nodes is sending a UDP packet to the root approximately every one minute.

[1] https://www.wireshark.org/

3 Link layer operation

To start with, open the provided PCAP file with Wireshark.

The wireshark window has three horizontal sections: Packet List (top); Packet Details (middle); Packet Bytes (bottom). They can be shown/hidden from the "View" menu. Make sure all 3 are visible.

See how selecting a header field under “Packet Details” will highlight the corresponding bytes in “Packet Bytes”. Remember this functionality for later.

3.1 Link layer broadcast: No ACKs

Focus on frame number 1 and in the packet details section expand the "IEEE 802.15.4" and "Frame Control Field" (FCF) sections.

- Observe how the FCF tells us the following:
    - This is a frame of type "Data".
    - The sender is not requesting an acknowledgment.
    - The frame version is 1.
    - The source address mode is extended/long (8 bytes) and the destination address mode is short (2 bytes).
- See the values of the remaining fields. Observe how:
    - The destination MAC address is `0xFFFF`. This is the broadcast address: It makes sense that the sender is not requesting an ACK! Remember: We never use ACKs for broadcast frames.
    - The source MAC address is `00:08:00:08:00:08:00:08`. This corresponds to device number 8 in the figure.
      See how Wireshark tries to resolve the OUI to a manufacturer name (Multitec). This is because `00:08:00` has been assigned to a manufacturer called Multitech Systems, Inc. Since this is a simulated network, the conversion from OUI to manufacturer is irrelevant, those MAC addresses are assigned to nodes by the simulator.

3.2 Link layer unicast. Data vs ACK frames

Now select frames number 10 and 11:

- Frame 10:
    - Observe how the destination address mode for frame 11 is extended/long (8 bytes). The destination address is unicast: `00:01:00:01:00:01:00:01`. This is the root's MAC address.
    - Observe how the sender is requesting an ACK.
    - Observe how the frame's sequence number is 140.
- Frame 11:
    - This is the ACK.
    - See how the FCF uses a different frame type (ACK)
    - See how the ACK itself specifies the sequence number of the frame being acknowledged; it matches the sequence number of frame 10.

Had frame 10 or the ACK been lost, the sender would re-transmit an exact copy of frame 10 using the same sequence number.

Question:

- *Can you think why the sequence number of the retransmission has to be the same?*
- *To answer, think of what can go wrong if the sender were to use a different sequence number.*

4 DODAG Formation with RPL

Now that you understand what is going on at layer two, let's focus on routing.

4.1 DODAG Information Solicitations (DIS)

Focus on frames 1 to 7: Wireshark should tell you that all those frames are of type "RPL Control: DODAG Information Solicitation". These are our DIS frames. As discussed in class, those frames are sent by devices to query about the presence of a nearby RPL network.

In the packet details window, expand the IPv6 section and observe:

- First look at the source IPv6 of those frames:
  - It is different for each frame.
  - It is always unicast, link-local.
  - Each of those source addresses corresponds to the sender's IPv6 address. For example, the source address of frame 1 is `fe80::208:8:8:8`. This is the IPv6 address of node number 8, which has been automatically generated with SLAAC. Observe the source address in the MAC header. Can you see the inverted U/L bit?
- Now look at the destination IPv6 address:
  - It is the same for all frames: `fe02::1a`. This is a well-known IPv6 address commonly known as "All RPL nodes". Every device that supports RPL must be willing to receive multicast packets sent to this address.
  - This address is multicast, link-local. Observe the destination address of the MAC header: Can you see how it is the short broadcast address (`0xFFFF`)?

4.2 "Upwards" route construction: DODAG Information Objects (DIO)

Focus on frame 8: See how wireshark tells you this is a DODAG Information Objects (DIO) packet. As discussed in class, this is used by nodes to advertise the presence of a network and its parameters.

Let's look inside this packet:

- First, observe how this particular packet has been sent by `fe80::201:1:1:1`. This is the root itself, but DIOs will be sent by all nodes once they have joined the network. Can you find any DIOs sent by other nodes? You should be able to find quite a few of them.
- See how the destination is, same as with DIS, the "All RPL nodes" address `ff02::1a` (link-local, multicast).

In the packet details window, expand the "Internet Control Message Protocol v6" section. This is something we did not discuss in-class: All RPL packets are carried inside ICMPv6 packets: This is a family of packets used to disseminate network control information, they are never used for user data. Ping also uses ICMP (for IPv4) or ICMPv6 packets.

In this ICMPv6 section, scroll down a bit and you will start seeing RPL-specific information. There is a lot of information here. Try to find the following:

- The RPL Instance ID.
- The RPL Version.
- The DODAG identifier. See how this is one of the root's universal/routable addresses.
- The root's rank.
- Prefix: When a device joins the network it will compute for itself a universal/routable IPv6 address using SLAAC. This will be the prefix!

4.2.1 Rank Calculation

Focus on frame 14. Device 4 has joined the network and is now in turn advertising the presence of a network using a DIO. Observe:

- Instance, DODAG ID, Version, Prefix are the same as in the DIO sent by the root.
- But the advertised rank is 512, higher than the rank of the root. Device 4 has calculated its own rank based on the quality of the link between itself and the root and also based on the rank advertised by the root.

Some devices, for example device 15, are too far away from the root and will have to rely on intermediate nodes for upwards routing. This can potentially lead to long paths.

Questions:

- *Find the sequence of DIOs used to construct the path from node 15 to the root.*
- *What is the path?*

4.2.2 Unicast DIOs: RPL Probing

If you scroll down the packet capture you will also be able to see DIOs with a unicast destination. These are used as part of a technique called “probing”, which is something we did not discuss in class (you don’t need to remember this for the exam).

Remember how each RPL device calculates its own rank using the rank of its preferred parent and the quality of the link between itself and the parent. Probing allows a sending device to quickly collect up-to-date information about the quality of the link between itself and the parent/DIO destination.

For example, frames 24-29 are probes and their respective ACKs. They have been sent by node 2 to estimate the quality of its link to the root.

4.3 “Downwards” route construction: Destination Advertisement Objects (DAO)

Focus on frame 10. This is device 4 advertising its presence to the root. Observe:

- DAOs use a <u>unicast</u> link-local destination address: They are sent to specific parents.
- Similar to DIS and DIO, they are ICMPv6 packets
- Their content has some similarities with DIO packets, but also many differences. One very important difference is the presence of the “RPL Target” section.

Questions:

- *If you revisit Figure 1 you will see how some distant devices (e.g. device 15) cannot communicate with the root directly. Downwards paths between the root and each of those devices are over multiple hops.*
- *Can you spot the sequence of DAOs used to construct the path from the root to node 15 and the root?*
- *What is the path?*
- *What can you observe about the “RPL Target” in all those DAOs?*
- *Can you write down the entry in the routing table for destination node 15 at each intermediate node of the path?*

5 6LoWPAN Compression with IPHC

The simulated network uses 6LoWPAN IP Header Compression exactly as discussed in class. In the Wireshark "Packet Data" section (bottom) you will see two tabs:

- Frame (N bytes): This is the packet as it went in the air
- Decompressed 6LoWPAN IPHC (M bytes): This tab shows the full decompressed headers. We can see this because Wireshark knows how to decompress 6LoWPAN IPHC.

See how the two byte counts are different. These are the savings from 6LoWPAN compression.

5.1 Stateless Compression (Link-Local): RPL Traffic

Focus on frame number 8. This is a DIO frame with the following characteristics:

- Source IPv6 address: Link-local, unicast: The address has been constructed using SLAAC on the node's MAC address. Check the source MAC address to verify this.
- Destination IPv6 address: Link-local, multicast.

Now expand the "6LoWPAN" section in the Packet Details panel. Observe the IPHC header:

- The hop limit is 64. This is a well-known value, so it can be removed from the packet and replaced with value 2 (binary 10) in the "Hop Limit" (HL) of the IPHC header.
- Source address:
  - Since it is link-local, SAC is stateless (SAC=0).
  - Since it has been generated using SLAAC, the destination can reconstruct it from the source MAC address. SAM=11 (binary) and the source IPv6 address is omitted entirely.
  - Select the "Source" line in the 6LoWPAN section. See how wireshark shows you the correct source address, but it does not highlight anything in the packet data panel at the bottom.
- Destination address:
  - It is multicast: M=1.
  - It is link-local, so DAC is also stateless (DAC=0).
  - The 120 most significant bits of the destination follow the pattern `ff02::00/120`. DAM=11 (binary)
  - There is no way the destination node can guess those 8 bits, so we have to carry them in the packet.
  - Select the "Destination" line in the 6LoWPAN section and see how wireshark will highlight the corresponding 8 bits in the packet data panel: All we had to include in the packet was the byte that the destination node cannot guess: `0x1a`.

Questions:

- *Find a frame with a link-local unicast destination.*
- *Without looking at the IPHC header, try to write down what you expect to see for M, DAC, DAM.*
- *Validate your answer by looking at the IPHC header.*
- *You are very likely looking at a frame where both source and destination IPv6 addresses have been elided in their entirety! How many bits did we save?*

5.2 Stateful Compression (Routable Addresses): UDP Data Traffic

Remember how all nodes are periodically sending a UDP packet to the root. Find any one of those UDP packets (e.g. frame 120).

Observe the following:

- Source: `::209:9:9:9`
- Destination: `::201:1:1:1`
- Wireshark is complaining that the packet's UDP checksum is incorrect.

There is clearly something wrong here. See how both addresses are missing the most significant part. See however how both of them could have been derived using SLAAC from the corresponding address of the MAC header. As discussed in class, for the construction of a routable address using SLAAC we need to somehow specify the network prefix. Nodes learn the prefix from DIO messages (Sec. 4.2). In our case this is `fd00::/64`.

For now, expand the IPHC section in "Packet Details" and observe:

- Source address:
  - SAC: Stateful (SAC=1)
  - SAM: Compressed. The destination node should be able to reconstruct the address by using "Context 0" for the network part (high 64 bits). The host part can be derived from the source MAC address.
  - There is something new here: The field "Source Context Identifier" with value 0. As discussed in class, the only way for stateful / context-based compression to work is if all devices across the network have been configured with the same contexts.
- Destination address:
  - Unicast, therefore M=0
  - Global/routable, therefore SAC: Stateful (DAC=1)
  - DAM: Compressed.
  - Destination Context: 0. We have the same problem.

In our simulated network devices have been administratively configured to use `fd00::/64` as Context 0. What happens here is that wireshark can reconstruct the device part of the source IPv6 address from the source MAC address, but it has no idea of knowing what "Context 0" is. This is what leads to all those errors.

We need to tell wireshark what contexts to use: Open wireshark's preferences ---> Protocols ---> 6LoWPAN and enter "`fd00::/64`" under "Context 0". See how all UDP packets now show the correct source/destination addresses and how the checksum errors have disappeared.

Question:

- *We fixed a problem with source/destination IPv6 addresses and that fixed UDP checksum calculation too. How is that possible? What do addresses have to do with checksum calculation?*

5.2.1 Stateful Compression Along Multiple Hops

Turn your attention to frames 162-169. These correspond to a UDP packet travelling from device 14 to the root, via 10, then 8, then 4. Thus, the full path is 14--->10--->8--->4--->1.

Observe:

- The source and destination IPv6 addresses of all those packets are the same across the entire path.
- But the source and destination MAC addresses change on a per-hop basis.

This means that at different hops the source/destination IPv6 address may or may not be correlated to the source/destination MAC address. This is a typical case of the same UDP/IPv6 datagram undergoing a different degree of compression at different hops:

- At the first hop (from device 14 to device 10), only the source IPv6 address is correlated to the corresponding MAC address.
- At the last hop (from device 4 to device 1), only the destination IPv6 address is correlated to the corresponding MAC address.
- No correlation at all at intermediate hops.
- Furthermore, the first hop uses "Hop Limit" of 64, which is a well-known value and can be compressed. However, at all subsequent hops the hop limit is not a well-known value and has to be carried in-line.

Questions:

- *Compare the packet sizes across those hops. What observation can you make?*
- *Compared the "Hop Limit" in the IPHC header across the hops.*
- *Inspect the address compression at all hops. Can you see how SAM/DAM vary and how different parts of the IPv6 addresses are carried inline?*