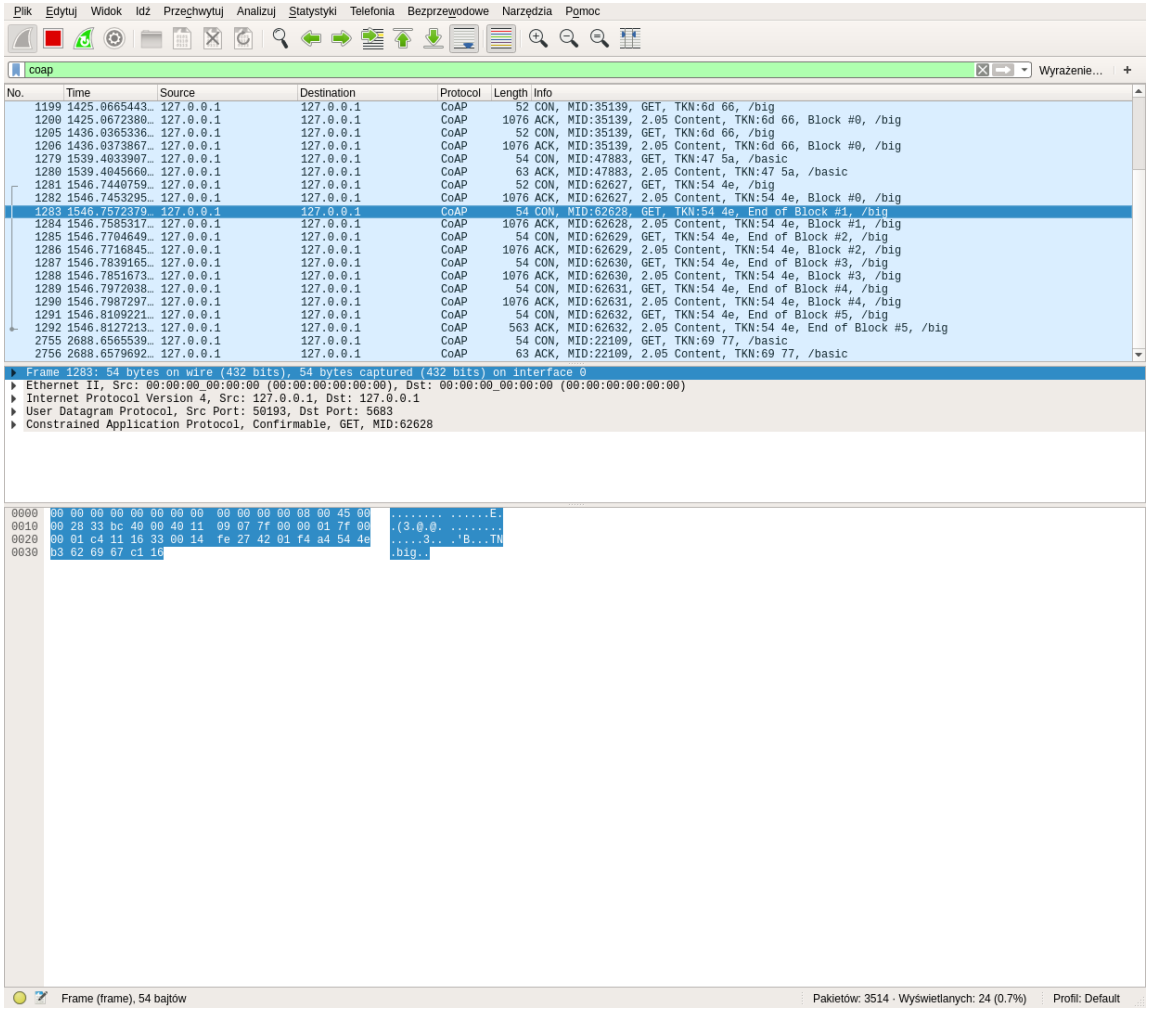
Przygotowanie do zajęć

Sklonuj repozytorium https://github.com/pwr-zak/iot-lab do swojego katalogu domowego. W repozytorium znajdziesz katalog „Zajecia_nr_5”, w którym znajdują się pliki potrzebne do wykonania zadań.

Celem dzisiejszych zajęć jest wykazanie i zrozumienie różnić w protokołach CoAP i HTTP i konsekwencji wynikających z wykorzystania UDP oraz TCP w ich implementacji.

Na każdym stanowisku w laboratorium zainstalowany został program Wireshark. Należy uruchomić go przed przystąpieniem do dalszej części zajęć. Praca z Wireshark powinna być możliwa bez konieczności uruchamiania go w kontekście super-użytkownika. Po uruchomieniu programu proszę rozpocząć nasłuchiwanie pracy interfejsu „loopback:lo”. Zarówno odwołania http jak i coap, udp czy tcp można filtrować w oknie aplikacji Wireshark, to skutecznie ułatwia odszukanie porządnych danych.



Ilustracja 1: Przykład listy filtrowanych transmisji protokołu CoAP

Ponad to, na każdym stanowisku zainstalowany został pakiet python’a CoAPthon, który jest implementacją w python 2.7 programów klienta i serwera pracujących wedle zasad protokołu CoAP. Po uruchomieniu serwera program zaczyna słuchać na porcie UDP:5683 oraz informuje o tym i domyślnie reaguje na URI, których listę wyświetla w terminalu:

```
coapserver.py

CoAP Server start on 0.0.0.0:5683
['/basic', '/storage', '/child', '/separate', '/etag', '/', '/big',
'/encoding', '/advancedSeparate', '/void', '/advanced', '/long', '/xml']
```

W celu odebranie wiadomości od serwera można uruchomić program klienta jak poniżej:

```
Command:    coapclient.py -o -p [-P]
Options:
      -o, --operation=  GET|PUT|POST|DELETE|DISCOVER|OBSERVE
      -p, --path=             Path of the request
      -P, --payload=          Payload of the request
      -f, --payload-file=           File with payload of the request
```

```
coapclient.py -o GET -p coap://127.0.0.1:5683/basic
2018-05-27 15:27:43,242 - MainThread - coapthon.layers.messagelayer -
DEBUG - send_request - From None, To ('127.0.0.1', 5683), None-None, GET-
Pf, [Uri-Path: basic, ] No payload
2018-05-27 15:27:43,242 - MainThread - coapthon.client.coap - DEBUG -
send_datagram - From None, To ('127.0.0.1', 5683), CON-53950, GET-Pf,
[Uri-Path: basic, ] No payload
2018-05-27 15:27:43,242 - Thread-1   - coapthon.client.coap - DEBUG -
Start receiver Thread
2018-05-27 15:27:43,243 - MainThread-Retry-53950 - coapthon.client.coap -
DEBUG - retransmit loop ... enter
2018-05-27 15:27:43,244 - Thread-1   - coapthon.client.coap - DEBUG -
receive_datagram - From ('127.0.0.1', 5683), To None, ACK-53950, CONTENT-
Pf, [] Basic Resource...14 bytes
2018-05-27 15:27:43,244 - Thread-1   - coapthon.layers.messagelayer -
DEBUG - receive_response - From ('127.0.0.1', 5683), To None, ACK-53950,
CONTENT-Pf, [] Basic Resource...14 bytes
2018-05-27 15:27:43,244 - Thread-1   - coapthon.client.coap - DEBUG -
Waiting for retransmit thread to finish ...
2018-05-27 15:27:43,246 - MainThread-Retry-53950 - coapthon.client.coap -
DEBUG - retransmit loop ... exit
Source: ('127.0.0.1', 5683)
Destination: None
Type: ACK
MID: 53950
Code: CONTENT
Token: Pf
Payload:
Basic Resource

2018-05-27 15:27:43,354 - Thread-1   - coapthon.client.coap - DEBUG -
Exiting receiver Thread due to request
```

W ramach instalacji python'a dostępny jest też dla Państwa mini serwer HTTP, z którego należy skorzystać w trakcie realizacji zadań przewidzianych na dzisiejszych zajęciach.

Przykład uruchomienia serwera HTTP nasłuchującego na interfejsie localhost na porcie 8880

```
usage: server.py [-h] [--cgi] [--bind ADDRESS] [port]

cd miniHTTP/
python3 -m http.server 8880
```

Serwer ten będzie traktował katalog roboczy w ramach którego uruchomiony został program server.py jako katalog root dla serwera HTTP.

Aby sprawdzić działanie serwera proszę uruchomić nowy wirtualny terminal (lub kolejną zakładkę w ramach wcześniej uruchomionej instancji) i wydać polecenie:

```
curl GET localhost:8880
```

W odpowiedzi w terminalu wyświetli się informacja jaką zwraca serwer HTTP np.:

```
curl: (6) Could not resolve host: GET
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML
4.01//EN"
"http://www.w3.org/TR/html4/strict.dtd">
<html>
<head>
<meta http-equiv="Content-Type"
content="text/html; charset=utf-8">
<title>Directory listing for /</title>
</head>
<body>
<h1>Directory listing for /</h1>
<hr>
<ul>
<li><a href="a">a</a></li>
<li><a href="basic">basic</a></li>
<li><a href="c">c</a></li>
</ul>
<hr>
</body>
</html>
```

Zadanie

Zastanów się i zaplanuj doświadczenie tak, by wykazać wyższość protokołu CoAP nad HTTP dla krótkich wiadomości, oraz że przy transmisji dużych paczek danych narzut związany z obsługą TCP może okazać się mimo wszystko mniej znaczący niż dalsze korzystanie z datagramów UDP.

1. Wygeneruj programem curl odpowiedź serwera HTTP i sprawdź w programie Wireshark ile bajtów zajmuje przesłanie minimalnej wiadomości jako odpowiedzi na GET,

   Pytanie, czy istnieje możliwość przesłania krótszej wiadomości?
   Jeśli tak, to ile bajtów potrzebne jest na jej przesłanie?

2. Podobnie jak w przypadku punktu 1 należy sprawdzić jaki rozmiar wyrażony bajtami wiadomości przesyłanej w ramach protokołu CoAP. W tym celu można posłużyć się programami coapserver.py oraz coapclient.py, lub napisać krótki program w języku python.

3. Spróbuj wykorzystując wiedzę i narzędzia z pierwszych dwóch etapów tego laboratorium wykazać, o ile efektywniejszy jest protokół CoAP w sytuacji transmisji minimalnej liczby danych od protokołu HTTP, wyraź to np. stosunkiem liczby bajtów na poziomie warstwy IP potrzebnych na przesłanie jednego bajtu informacji w ramach badanych protokołów warstwy aplikacji.

4. Jaki jest minimalny progowy rozmiar paczki w TCP i transmisji protokołem HTTP kiedy CoAP przestaje być opłacalny; przyjmij za kryterium oceny liczbę bajtów potrzebną do przesłania pełnej wiadomości podobnie jak w punkcie 3. Zbierz potrzebne dane do pliku i spróbuj wykreślić wykres aby zilustrować szukaną zależność odpowiednio dla opcji CoAP:
    a) z wymuszonym potwierdzeniem transmisji,
    b) bez potwierdzenia ACK.

Do realizacji punktu 3 tego zadania proszę skorzystać z programu gnuplot lub z funkcji python matplotlib.

Dokumentacja

- *https://tools.ietf.org/html/rfc7252*
- *http://coap.technology/*
- *https://github.com/Tanganelli/CoAPthon*
- *https://en.wikipedia.org/wiki/User_Datagram_Protocol*
- *https://en.wikipedia.org/wiki/Hypertext_Transfer_Protocol*
- *https://en.wikipedia.org/wiki/Transmission_Control_Protocol*
- *https://developer.mozilla.org/en-US/docs/Web/HTTP/Basics_of_HTTP/MIME_types*
- *https://en.wikipedia.org/wiki/Uniform_Resource_Identifier*
- *http://www.gnuplot.info/*
- *https://matplotlib.org/users/pyplot_tutorial.html*