Obiekty Internetu Rzeczy

(Laboratorium 1)

Krzysztof Pierczyk

listopad 2020

1 Ćwiczenie 1: Interakcja z serwerem CoAP

Pierwsze z zadań miało na celu zapoznanie się funkcjonalnością wtyczki **Copper** umożliwiającej komunikację za pośrednictwem protokołu CoAP z poziomu przeglądarki internetowej. Niestety wtyczka nie jest wspierana w aktualnej wersji Firefox, dlatego wymagana była instalacja starszej edycji. Wydanie dostarczone w ramach przedmiotu zostało przygotowane dla platformy Windows. Ze względu na korzystanie z dystrybucji Linuxa konieczne było ręczne pobranie odpowiedniej wersji przeglądarki. Wybór padł na ostatnią ze wspieranych przez wtyczkę edycji, która umożliwiała uruchomianie nieautoryzowanych dodatków (52.9.0esr). Kolejnym krokiem było uruchomienie serwera CoAP na platformie docelowej. W związku z realizacją ćwiczenia na innej niż domyślnie płytce ewaluacyjnej (tj. NodeMCU z układem SoC ESP8266) konieczne było przygotowanie przykładowej implementacji. Wykorzystano w tym celu platformę ESP-SDK oraz otwartoźródłową bibliotekę **libcoap**.

Przygotowana aplikacja łączy się poprzez WiFi z uprzednio skonfigurowanym punktem dostępu i wykorzystując protokół DHCP uzyskuje adres IP. Następnie inicjalizuje zasoby i rozpoczyna nasłuchiwanie na domyślnym porcie CoAP (5683). Serwer udostępnia dwa zasoby: /**time** oraz /**colour**. Na pierwszym z nich możliwe jest wykonanie jedynie metody GET. Zawiera on aktualną datę oraz godzinę udostępnianą w postaci tekstu ASCII. Czas jest synchronizowany z wykorzystaniem protokołu SNTP (ang. *Simple Network Time Protocol*). Zasób /colour stanowi tryplet 8-bitowych liczb całkowitych symbolizujących składowe RGB pewnego koloru. Na tym zasobie możliwe jest wykonanie zarówno metody GET jak i PUT. Serwer udostepnia również abstrakcyjny zasób /**.well-known/core** zgodny z RFC5786. Po wykonaniu na nim metody GET, klient otrzymuje przykładowe informacje o pozostałych zasobach.

```
0020  00 5e 16 33 aa fc 00 51  0a 95 60 45 8c bd c1 28   ·^·3···Q ··`E···(
0030  b1 12 ff 20 25 64 20 25  64 22 3b 69 66 3d 22 47   ··· %d % d";if="G
0040  45 54 20 50 55 54 22 3b  72 74 3d 22 63 6f 6c 6f   ET PUT"; rt="colo
0050  75 72 22 3b 63 74 3d 22  70 6c 61 69 6e 20 74 65   ur";ct=" plain te
0060  78 74 22 3b 6f 62 73 6f  75 72 3e 3b 70 75 74 3d   xt";obso ur>;put=
0070  22 25 64                                          "%d
```

Rysunek 1: Drugi blok odpowiedzi na zapytanie 'GET /.well-known/core'

W ramach testów wykonano serię zapytań do serwera rejestrując przy tym ruch sieciowy za pomocą narzędzia **tshark**. Plik wyjściowy (`ex_1.pcapng`) został dołączony do sprawozdania. Po jego uruchomieniu w programie **wireshark** możliwe jest zweryfikowanie poprawności odpowiedzi wysyłanych przez serwer. Można tu zauważyć, że wtyczka Copper domyślnie przesyła zapytania GET z ustawioną opcją *Block2* oraz wielkością bloku równą 64. Pozostałe cechy przesyłanych pakietów były zgodne z oczekiwaniami. Warto zwrócić jednak uwagę na drugi blok odpowiedzi wysłanej w zwiazku z zapytaniem o zasób `/.well-known/core` (Rys. 1). Na jego końcu znajduje się (zaznaczony na czerwono) losowy ciąg znaków. Krótkie śledztwo pozwoliło ustalić, iż jest to błąd jednej funkcji z biblioteki libcoap, która wysyła w tym przypadku blok określonej długosci (tu: 64) bez względu na faktyczną ilość

danych pozostałych do wysłania. Sytuacja taka nie zachodzi przy okazji zapytań o pozostałe zasoby. Zrezygnowanie z wczesnej negocjacji rozmiaru bloku eliminuje ten problem.

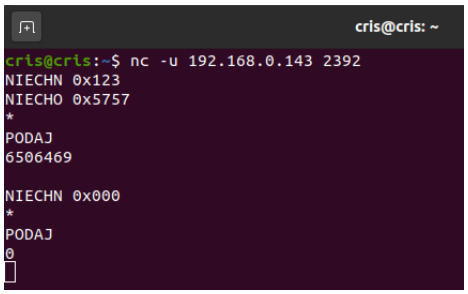
2 Ćwiczenie: przykładowy serwer UDP

W drugim zadaniu utworzony został serwer UDP nasłuchujący na porcie 2392. Środowisko ESP-SDK implementuje API gniazd sieciowych zgodnych z gniazdami BSD, przez co konfiguracja przebiegła podobnie jak na standardowej maszynie klasy PC. Jedynym zadaniem serwera było odwrócenie kolejności znaków w otrzymanym datagramie i odesłanie go w takiej postaci. Z ciekawostek można dodać, że wykorzystano w tym procesie algorytm XOR. Faza testów obejmowała przesłanie kilku przykładowych wiadomości przy użyciu systemowego narzędzia **netcat**. Zarejestrowany ruch sieciowy został załączony do sprawozdania (`ex_2.pcapng`).

Rysunek 2: Przykładowa odpowiedź serwera UDP

3 Ćwiczenie 3: prosty kalkulator z intefejsem sieciowym

Ostatnie z zadań obejmowało zmodyfikowanie serwera utworzonego w ćwiczeniu drugim. Miał on zostać przygotowany do przyjęcia jednej z czterech komend: *NIECHN [hex-number]*, *NIECHO [hex-number]*, * oraz *PODAJ*. Piewsze dwie ustawiają wartość zmiennych przechowywanych na serwerze. Kolejna powoduje wykonanie mnożenia aktualnych wartości oraz zachowanie wyniku w osobnej zmiennej. Ostatnie polecenie skutkuje wysłaniem do klienta datagramu z ostatnim zapisanym wynikiem mnożenia. Rys. 3 przedstawia przykładowy wynik interakcji z serwerem. Zarejestrowany ruch sieciowy ponownie został dołączony do sprawozdania (`ex_3.pcapng`). Treść wysyłanych do serwera w ramach realizacji scenariusza testowego to kolejno: PODAJ, NIECHN 0x576, NIECHO 0x298, *, PODAJ, NIECHN 0x311, NIECHO 0x987, *, PODAJ. Odpowiedzi serwera zawierały poprawne wyniki operacji.



Rysunek 3: Przykładowe działania wykonane na zlecenie klienta