Systemy rozproszone | Technologie middleware, cz. II

Łukasz Czekierda, Katedra Informatyki AGH (luke@agh.edu.pl)

1. Przygotowanie do zajęć i weryfikacja środowiska

Co będzie potrzebne:

- Java (sugerowana wersja 8)
- IDE: Eclipse, IntelliJ
- wireshark z możliwością przechwytywania pakietów przechodzących przez interfejs loopback (jak na poprzednich zajęciach), alternatywnie: Wireshark bez takiej możliwości i dwa komputery mogące się wzajemnie komunikować
- kompilator Google Protocol Buffers (proto)
- plugin gRPC do kompilatora proto

Weryfikacja czy wszystko jest gotowe na zajęcia:

- Poprawne wykonanie komendy (wersja dla Windows): **protoc.exe -I=. --java_out=gen --plugin=protoc-gen-grpc-java=protoc-gen-grpc-java.exe --grpc-java_out=gen test.proto** *(może być konieczne wskazanie ścieżki plików .exe)*

2. Wykonanie ćwiczenia

2.1 Wprowadzenie

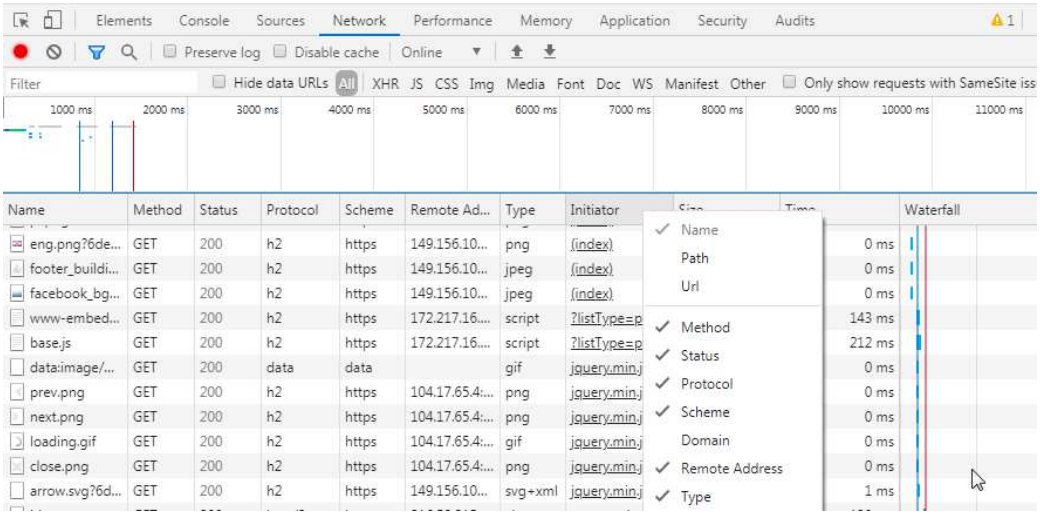
W miarę możliwości technicznych Prowadzący wprowadzi Studentów w temat ćwiczenia sprawdzając równocześnie ich przygotowanie. W razie braku takich możliwości, Student sam zapoznaje się z udostępnioną prezentacją.

2.2 Podstawy protokołu HTTP/2

1) Włącz **wireshark** (najlepiej w trybie *non-promiscuous*). Dodaj filtr **ip.addr==149.156.100.0/24.**
2) Używając przeglądarki **Chrome** otwórz stronę **http://www.informatyka.agh.edu.pl**. Dlaczego finalnie został użyty protokół HTTPS? Która ze stron zażądała zmiany i w jaki sposób?
3) Przeanalizuj ustanawianie komunikacji TLS (włącz filtr **ip.addr==149.156.100.0/24 && ssl** w wireshark): szukaj pola **Client Hello** a w nim **Application Layer Protocol Negotiation**. Co tu jest negocjowane?

```
⊞ Extension: SessionTicket TLS
⊟ Extension: Application Layer Protocol Negotiation
    Type: Application Layer Protocol Negotiation (0x0010)
    Length: 14
    ALPN Extension Length: 12
  ⊟ ALPN Protocol
      ALPN string length: 2
      ALPN Next Protocol: h2
      ALPN string length: 8
      ALPN Next Protocol: http/1.1
⊞ Extension: status_request
```

4) Aktywuj podgląd komunikacji w przeglądarce (**Ctrl-Shift-J)** i włącz prezentację wartości pól zaznaczonych na poniższym zrzucie ekranu (lista aktywna po kliknięciu na wiersz Name-Method…)



5) Ponownie załaduj stronę WWW i sprawdź, która wersja protokołu HTTP jest wykorzystywana.
6) Znajdź dwie strony WWW, dla których obsługi działa protokół HTTP/2 i dwie, dla których jest używane HTTP/1.1. Czy główna strona AGH obsługuje HTTP/2? Użyj np. **https://http2.pro** lub sprawdź sam(a).

2.3 Serializacja Google Protocol Buffers

1. Otwórz plik **person.proto** i zapoznaj się z jego zawartością.
2. Skompiluj plik z definicją interfejsu: otwórz okno konsolowe i z poziomu głównego katalogu projektu wydaj polecenie (wersja dla Windows) **protoc.exe -I=. --java_out=gen person.proto.** Ze względu na poprawność kompilacji całości projektu, wykonaj także kompilację opisaną w punktach 2.4.3 i 2.4.10.
3. Zapoznaj się z wygenerowanymi plikami.
4. Skompiluj plik ponownie żądając generacji kodu dla wybranych innych języków programowania **(--ruby_out**, **--python_out**, --**cpp_out**, …).
5. Sprawdź ile bajtów zajmuje przykładowa wiadomość serializowana w metodzie **ProtoSerialization.testProto()** – skonwertuj ją do tablicy bajtów: **.toByteArray().** Dodatkowo wypisz ją na konsolę korzystając z prezentowanej w kodzie konwersji do łańcucha znaków.
6. Sprawdź zgrubnie czas serializacji pojedynczej przykładowej wiadomości tego typu wykonując w pętli odpowiednio dużą liczbę serializacji – tak, by dało się wyznaczyć czas trwania pojedynczej.
7. Porównaj czas i efektywność (wielkość) serializacji Protocol Buffers z domyślną serializacją Java – kod w pliku **JavaSerialization.java**.
8. Dodaj do definicji **person.proto** nową (dowolną) wiadomość przesyłającą sekwencję liczb niecałkowitych (oznaczającą np. wysokość przychodów osoby w ostatnich miesiącach). Użyj słowa kluczowego **repeated**. Ponownie skompiluj i przeprowadź serializację nowej wersji wiadomości zawierającej np. trzy liczby w sekwencji. O ile zwiększyła się długość wiadomości?

2.4 Google RPC

1. Zaimportuj projekt do IDE.
2. **Analiza interfejsu.** Zapoznaj się z definicją interfejsu zawartą w pliku **calculator.proto**. Zawiera on nie tylko definicję wiadomości, ale i …
3. **Kompilacja definicji interfejsu.** Skompiluj plik z definicją interfejsu: otwórz okno konsolowe (MS-DOS) i z poziomu głównego katalogu projektu wydaj polecenie (wersja dla Windows) **protoc.exe -I=. --java_out=gen --plugin=protoc-gen-grpc-java=protoc-gen-grpc-java.exe --grpc-java_out=gen calculator.proto**
4. Jeśli IDE nie realizuje automatycznego odświeżania w razie zmian zawartości projektu na dysku, wymuś jego odświeżenie. Występujące wcześniej błędy kompilacji powinny zniknąć. W przypadku korzystania z **IntelliJ**, po kompilacji interfejsu oznacz folder **gen** jako **Generated Sources Root**.
5. **Analiza kodu.** Przeanalizuj wygenerowane pliki źródłowe. Zaobserwuj m.in. sposób pozyskania referencji do zdalnego obiektu w aplikacji klienckiej.
6. **Uruchomienie aplikacji.** Uruchom klienta i serwer oraz przetestuj poprawność działania aplikacji.
7. **Analiza komunikacji sieciowej.** Prześledź komunikację pomiędzy klientem i serwerem korzystając z wireshark. Jaki protokół komunikacji jest wykorzystany? Włącz w wireshark odpowiednie dekodowanie tego protokołu (**decode as…)**
8. **Rozbudowa interfejsu.** Do interfejsu **Calculator** dodaj nową operację mnożącą N liczb i zwracającą ich iloczyn. Zaimplementuj ją i przetestuj działanie aplikacji.
9. **Podejście obiektowe czy usługowe?** Zaobserwuj (testując), czy jest możliwe udostępnienie dla zdalnych wywołań kilku obiektów implementujących a) ten sam b) różne interfejsy IDL naraz - rozbudowując serwer by obsługiwał kolejną usługę przez dodanie w jego kodzie kolejnego **.addService**.
10. **Kompilacja definicji interfejsu.** Zapoznaj się z zawartością pliku **streaming.proto** i skompiluj go analogicznie jak poprzednio.
11. **Strumieniowanie przez serwer (server-side).** Wywołaj operację **generatePrimeNumbers** - zaobserwuj strumieniowanie. Narysuj diagram interakcji HTTP/2 pomiędzy klientem i serwerem. Czy to podejście ułatwia prowadzenie komunikacji w środowiskach gdzie klient jest „za NATem”?
12. **Strumieniowanie przez klienta (client-side).** Wywołaj operację **countPrimeNumbers** - zaobserwuj strumieniowanie. Narysuj diagram interakcji HTTP/2 pomiędzy klientem i serwerem. Czy to podejście ułatwia prowadzenie komunikacji w środowiskach gdzie klient jest „za NATem”?