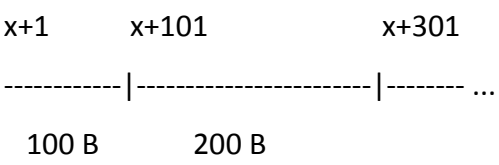
Funkcje protokołu TCP (powtórzenie z wykładu)

- dostarczenie danych do aplikacji określonej w polu „port docelowy”
- zapewnienie niezawodnej transmisji danych; segmenty docierają wszystkie, bez powtórzeń i w takiej kolejności w jakiej zostały wysłane
  (numerowanie segmentów w polu SeqNo i potwierdzanie ich odbioru w polu AckNo)
- zapewnienie integralności danych; dane docierają do odbiorcy w niezmienionej postaci
  (pole suma kontrolna)
- kontrola przepływu (ang. flow control), czyli dostosowanie tempa nadawania do tempa przetwarzania odbiorcy
  (pole Window informujące, ile jest wolnego miejsca w buforze odbiorcy)

Protokół TCP realizuje tzw. transmisję strumieniową, tzn. od nadawcy do odbiorcy płynie strumień danych podzielony na segmenty. Strumień to przesyłany w sieci ciąg bajtów pewnego zbioru danych, w którym nie ma ubytków, powtórzeń ani zmiany kolejności.
Strumień jest identyfikowany losowo generowanym numerem x.
Pierwszy bajt w strumieniu ma numer x+1.
Segmenty nie są numerowane kolejnymi liczbami całkowitymi, tylko numerami pierwszych bajtów w kolejnych segmentach, tak jak na rysunku:



SeqNo(1) = x+1, SeqNo(2) = x+101, SeqNo(3) = x+301, ...
1,2,… - numery porządkowe kolejnych segmentów
SeqNo(1), SeqNo(2),… - numery sekwencyjne kolejnych segmentów

Reguła numerowania segmentów:

Nr sekw. segmentu n+1 = Nr sekw. segmentu n + Liczba bajtów danych w segmencie n

SeqNo(n+1) = SeqNo(n) + LB(n), gdzie $n \geq 1$

Reguła potwierdzania segmentów:

Odebranie segmentu o numerze porządkowym n jest potwierdzane przez wysłanie w polu AckNo numeru sekwencyjnego (SeqNo) segmentu o numerze porządkowym n+1, czyli AckNo(n) = SeqNo(n+1)

Obowiązuje zasada kumulacji potwierdzeń, tzn. potwierdzając odbiór segmentu n odbiorca potwierdza zarazem odbiór wszystkich poprzednich n–1 segmentów.

Obserwacja ruchu sieciowego generowanego przez aplikację echo działającą na bazie TCP

Uruchamianie serwera aplikacji echo (Linux):

systemctl status iptables <- sprawdzenie czy mechanizm iptables jest aktywny

systemctl stop iptables <- wyłączenie iptables

systemctl status xinetd <- sprawdzenie czy jest uruchomiony xinetd

systemctl start xinetd <- uruchamianie xinetd (tylko jeśli powyższe polecenie wykaże nieaktywność)

cd /etc/xinetd.d <- przejście do katalogu z plikami konfig. usług kontrolowanych przez xinetd

ls <- wypisanie zawartości katalogu (bez plików ukrytych)

less echo-stream <- sprawdzanie czy serwer usługi echo jest aktywny (disable = no)

Jeśli serwer echo jest nieaktywny (disable = yes), to trzeba zmienić yes na no

i przeładować xinetd poleceniem systemctl reload xinetd.

Przygotowanie Wireshark do przechwytywania ramek aplikacji echo:

filtr przechwytywania: host <adres klienta> and host <adres serwera> and tcp port 7

włączenie przechwytywania: Capture -> Start

Uruchamianie klienta echo i wysyłanie danych z systemu Windows do serwera (Linux):

PuTTY -> wpisać adres serwera, Port 7, Connection Type Raw, Open

Wpisać krótki tekst (ala ma kota) i wcisnąć Enter

Tekst zostaje powtórzony w następnej linii

Zamknąć okno PuTTY

Wireshark powinien przechwycić 12 ramek

Analiza przechwyconych ramek:

Etap 1 - otwieranie (wirtualnego) połączenia TCP:

1. Klient żąda otwarcia połączenia TCP

   klient -> serwer: SrcPort=<losowy>, DstPort=7, SYN, SeqNo=x

   (x to identyfikator strumienia danych wysyłanych od klienta do serwera)

2. Serwer wyraża zgodę na otwarcie połączenia TCP

   serwer -> klient: SrcPort=7, DstPort=<losowy>, SYN, SeqNo=y, ACK, AckNo=x+1

   (y to identyfikator strumienia danych wysyłanych od serwera do klienta)

3. Klient potwierdza odebranie zgody

   klient -> serwer: SeqNo=x+1, ACK, AckNo=y+1

Uwaga: Odebranie segmentu z flagą SYN jest potwierdzane przez wysłanie w polu AckNo
wartości SeqNo(segment potwierdzany) + 1

Etap 2 - utrzymywanie połączenia/transmisja danych:

4. Klient wysyła 1-szą porcję danych do serwera:

   klient -> serwer: SeqNo=x+1, 11 bajtów danych, ACK, AckNo=y+1

5. Klient wysyła Enter (0x0d0a) do serwera:

   klient -> serwer: SeqNo=x+12, 2 bajty danych, ACK, AckNo=y+1

6. Serwer potwierdza odebranie 1-szej porcji danych od klienta

   serwer -> klient: SeqNo=y+1, 0 bajtów danych, ACK, AckNo=x+12

7. Serwer potwierdza odebranie Enter od klienta

   serwer -> klient: SeqNo=y+1, 0 bajtów danych, ACK, AckNo=x+14

8. Serwer wysyła echo 1-szej porcji danych i Enter do klienta

   serwer -> klient: SeqNo=y+1, 13 bajtów danych, ACK, AckNo=x+14

9. Klient potwierdza odebranie echa od serwera

klient -> serwer: SeqNo=x+14, 0 bajtów danych, ACK, AckNo=y+14

W etapie transmisji danych numery sekwencyjne segmentów nie są kolejnymi liczbami całkowitymi, tylko numerami pierwszych bajtów kolejnych segmentów,
więc numer sekwencyjny następnego segmentu to numer sekwencyjny poprzedniego plus liczba bajtów danych w poprzednim segmencie, czyli SeqNo(n+1)=SeqNo(n)+LB(n)

Potwierdzenie odebrania segmentu od nadawcy polega na wysłaniu w polu AckNo numeru sekwencyjnego następnego segmentu spodziewanego ze strony nadawcy.
Można to zapisać następująco:
AckNo(n) = SeqNo(n+1),
gdzie n oznacza n-ty odebrany segment, AckNo(n) oznacza numer wysyłany w polu AckNo dla potwierdzenia odbioru n-tego segmentu, SeqNo(n+1) oznacza numer sekwencyjny jeszcze nieodebranego (n+1)-go segmentu.

Zamykanie połączenia:

10. Klient wysyła żądanie zamknięcia połączenia:

klient -> serwer: SeqNo=x+14, 0 bajtów danych, FIN, ACK, AckNo=y+14

11. Serwer potwierdza odebranie żądania zamknięcia połączenia i akceptuje je:

serwer -> klient: SeqNo=y+14, 0 bajtów danych, FIN, ACK, AckNo=x+15

12. Klient potwierdza odebranie akceptacji

klient -> serwer: SeqNo=x+15, 0 bajtów danych, ACK, AckNo=y+15

Uwaga: Odebranie segmentu z flagą FIN jest potwierdzane przez wysłanie w polu AckNo wartości SeqNo(segment potwierdzany) + LB(segment potwierdzany) + 1
(tak samo jest potwierdzane odebranie segmentu z flagą SYN)