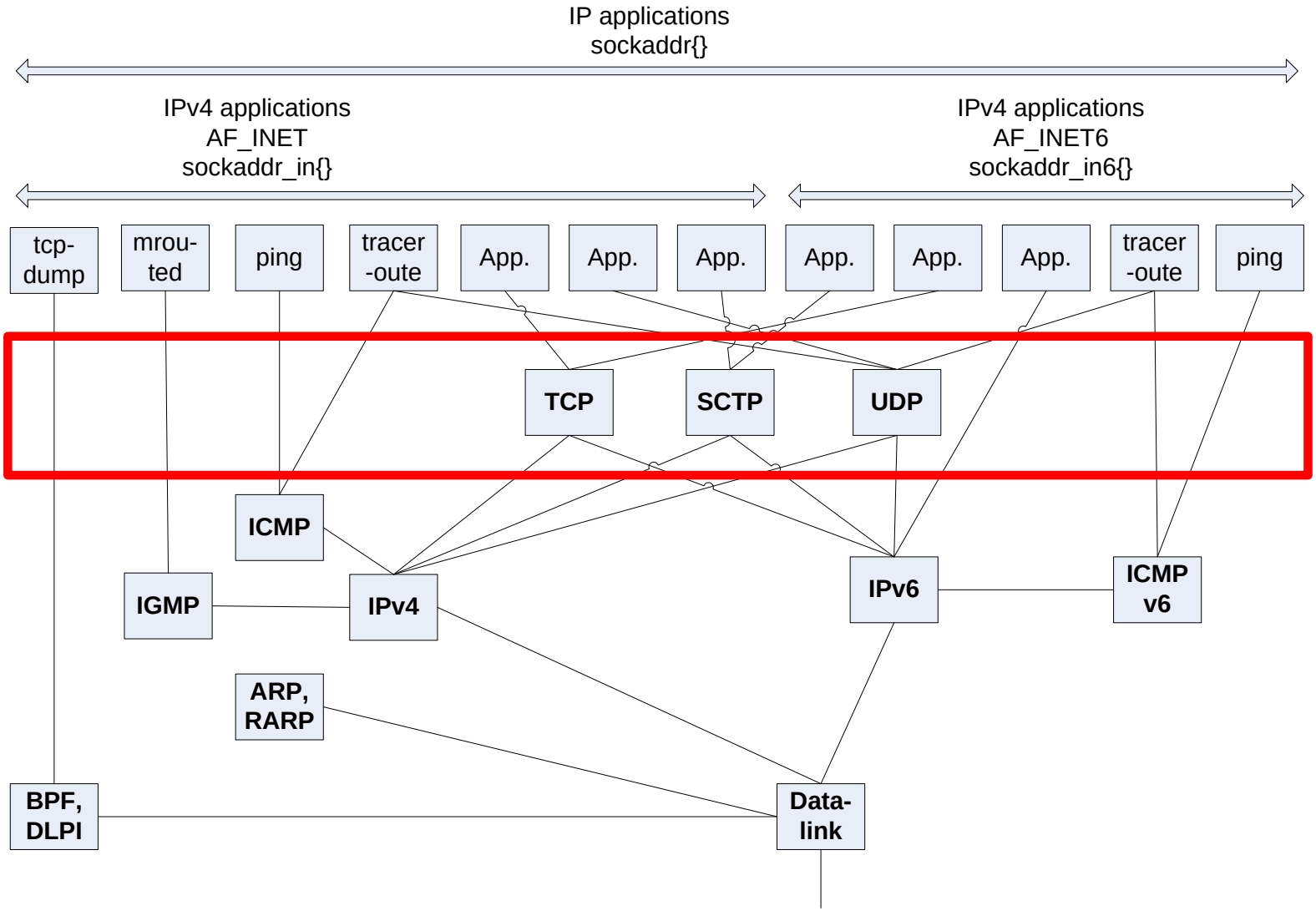
Wykład #2
Warstwa transportowa

Protokoły transportowe: UDP, TCP, SCTP. Protokół TCP. Rozmiary buforów i ograniczenia. Opcje gniazd.

LAB01 - podsumowanie

	ifconfig	**netstat**	**ip**	**route**
Odczyt adresu L3 (IP)	**+**	**+**	**+**	
Ustawienie adresu L3 (IP)	**+**		**+**	
Odczyt adresu L2 (MAC)	**+**	**+**	**+**	
Ustawienie adresu L2 (MAC)	**+**		**+**	
Odczyt tablicy rutingu		**+**	**+**	**+**
Zmiana tablicy rutingu			**+**	**+**
Liczba i nazwy interfejsów sieciowych	**+**	**+**	**+**	
Stany gniazd (**ss, lsof**)		**+**		
Liczba i nazwy interfejsów fizycznych	**lspci, lsusb**			

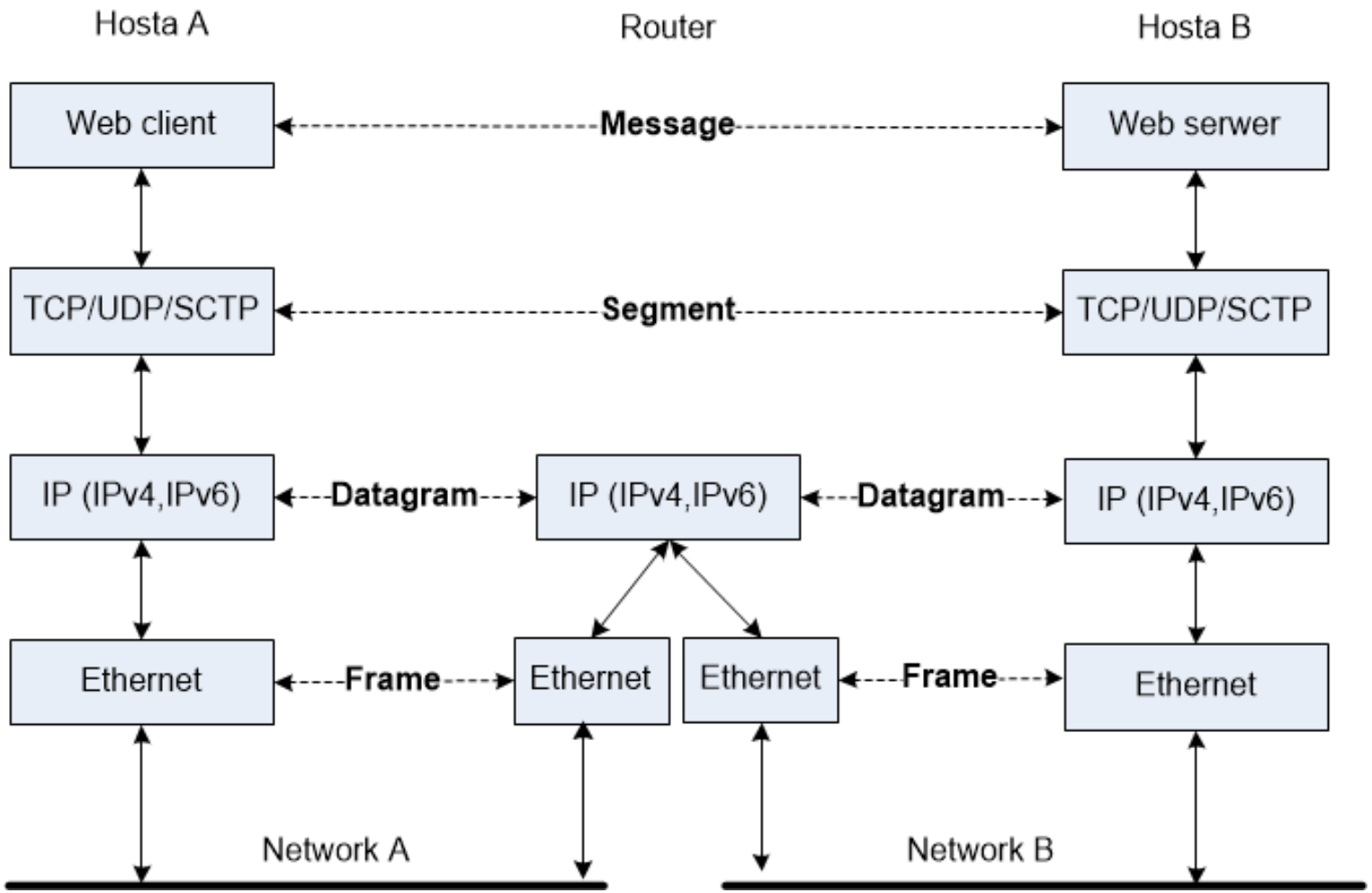
Ogólny przegląd protokołów sieci TCP/IP



Zastosowanie protokołów sieci TCP/IP

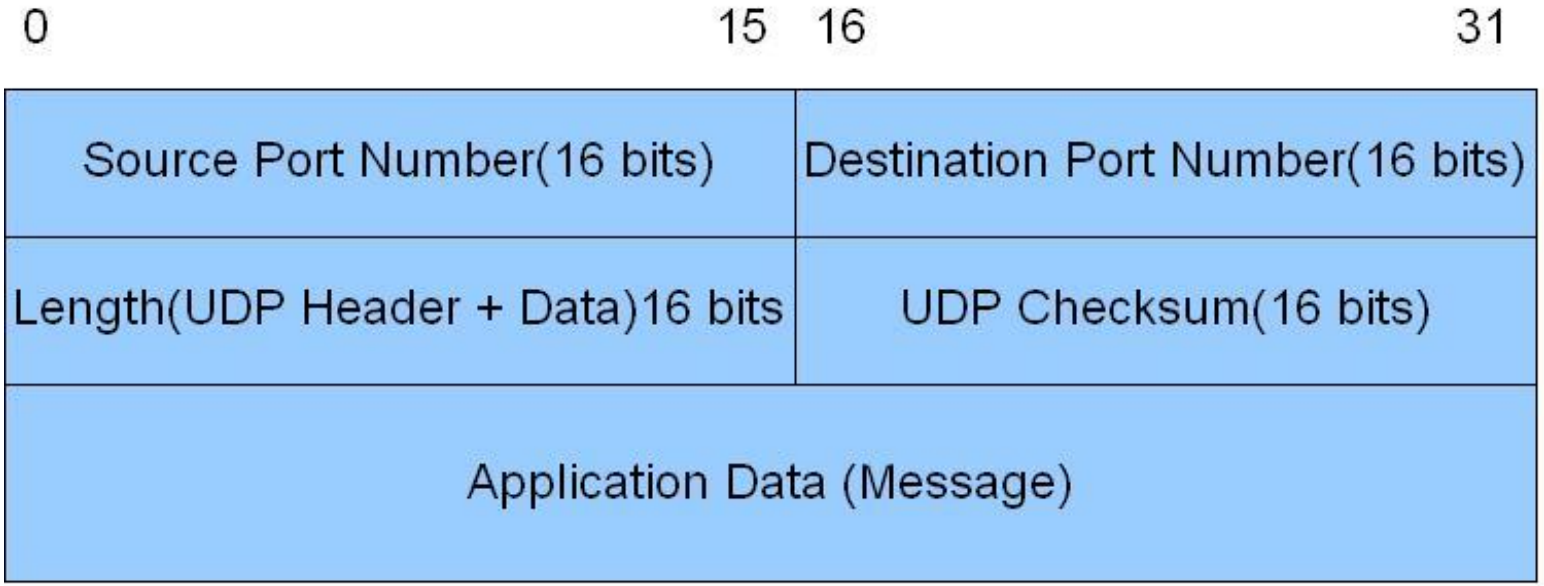
Application	IP	ICMP	UDP	TCP	SCTP
Ping		•			
Traceroute/tracepath		•	•		
OSPF (routing protocol)	•				
RIP (routing protocol)			•		
BGP (routing protocol)				•	
BOOPT (bootstrap protocol)			•		
DHCP (bootstrap protocol)			•		
NTP (time protocol)			•		
TFTP			•		
SNMP (managment)			•		
SMPT (mail)				•	
Telnet (remote login)				•	
SSH (secure remote login)				•	
FTP (file transfer)				•	
HTTP (WWW)				•	
NNTP (news)				•	
LPR (remote printing)				•	
DNS			•	•	
NFS			•	•	
SUN RPC			•	•	
DCE RPC			•	•	
IUA (SS7)					•
M2UA, M3UA (SS7)			•	•	•
H.248 (media gateway control)			•	•	•
H.323 (IP telephony)			•	•	•
SIP (IP telephony)					•
LTE (signalling)					•

Warstwa transportowa w sieci TCP/IP – wpływ na funkcjonalność aplikacji

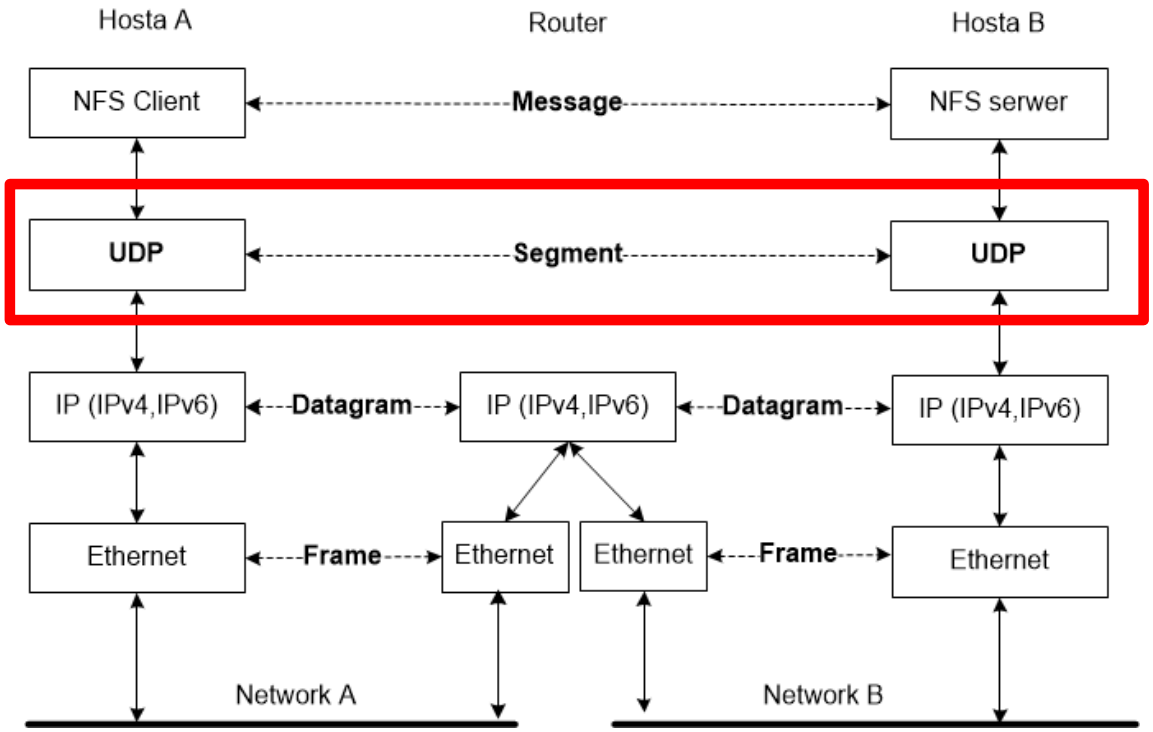


Protokół *User Datagram Protocol* (UDP)

- **RFC 768** (3 strony)
- Usługa bezpołączeniowa
- Przesyłanie datagramów – dane podzielone na paczki danych
- Zawodny – możliwa utrata pakietów bez żadnej informacji
- **Na jednym gnieździe możliwa jednoczesna komunikacja z wieloma systemami zdalnymi**



Warstwa transportowa UDP w sieci TCP/IP wpływ na funkcjonalność aplikacji

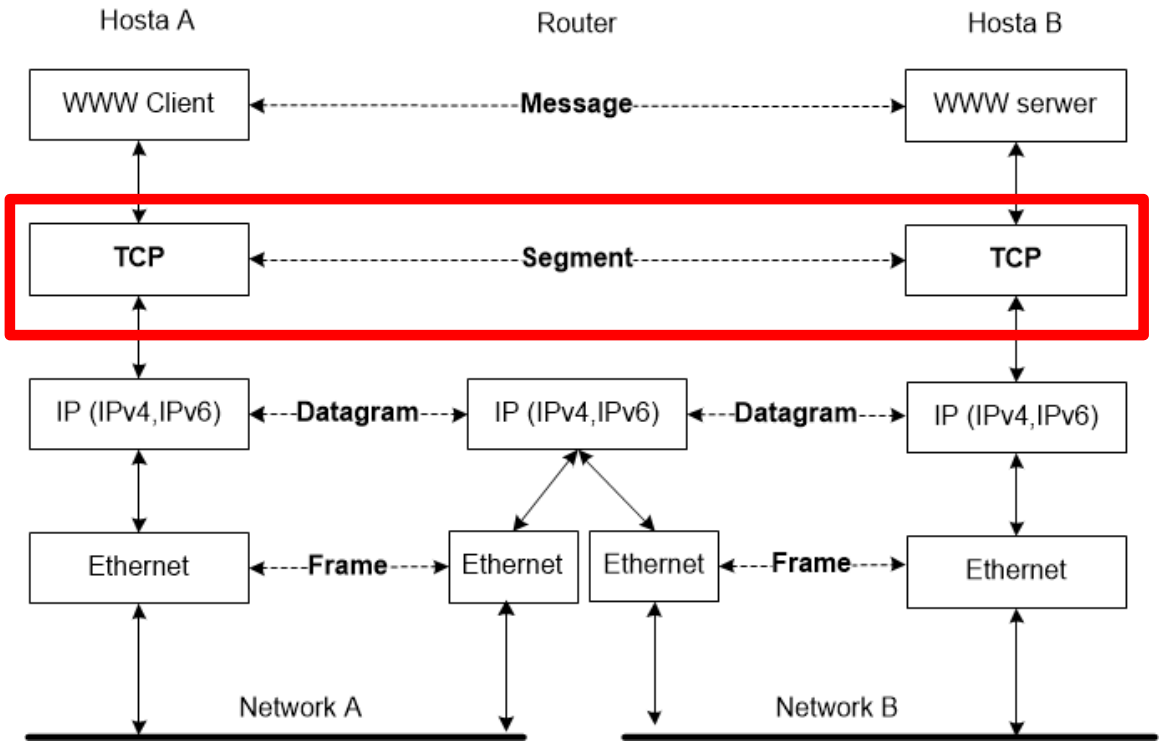


- Dostarcza mechanizmu adresowania aplikacji przez porty
- Funkcjonalnie taka sama usługa jak dla protokołu sieciowego IP
- Aplikacja musi zadbać o zagubione i dostarczone w złej kolejności pakiety
- Brak sterowania przepływem
- Brak wykrywania przeciążeń w sieci
- Obsługuje adresy typu **unicast, multicast i broadcast**

Protokół *Transmission Control Protocol* (TCP)

- **RFC 793**
- Protokół połączeniowy – należy nawiązać połączenie przed wymianą danych
- Zapewnia niezawodność – potwierdzenia, automatyczne ponawianie wysyłania danych, szacowanie RTT
- Porządkowanie danych – zachowanie kolejności bajtów w strumieniu danych
- Sterowanie przepływem - zapobieganie przeciążeniom
- **Na pojedynczym gnieździe komunikacja tylko dla jednego połączenia**
- Zorientowany na przesył bajtów danych

Warstwa transportowa TCP w sieci TCP/IP - wpływ na funkcjonalność aplikacji

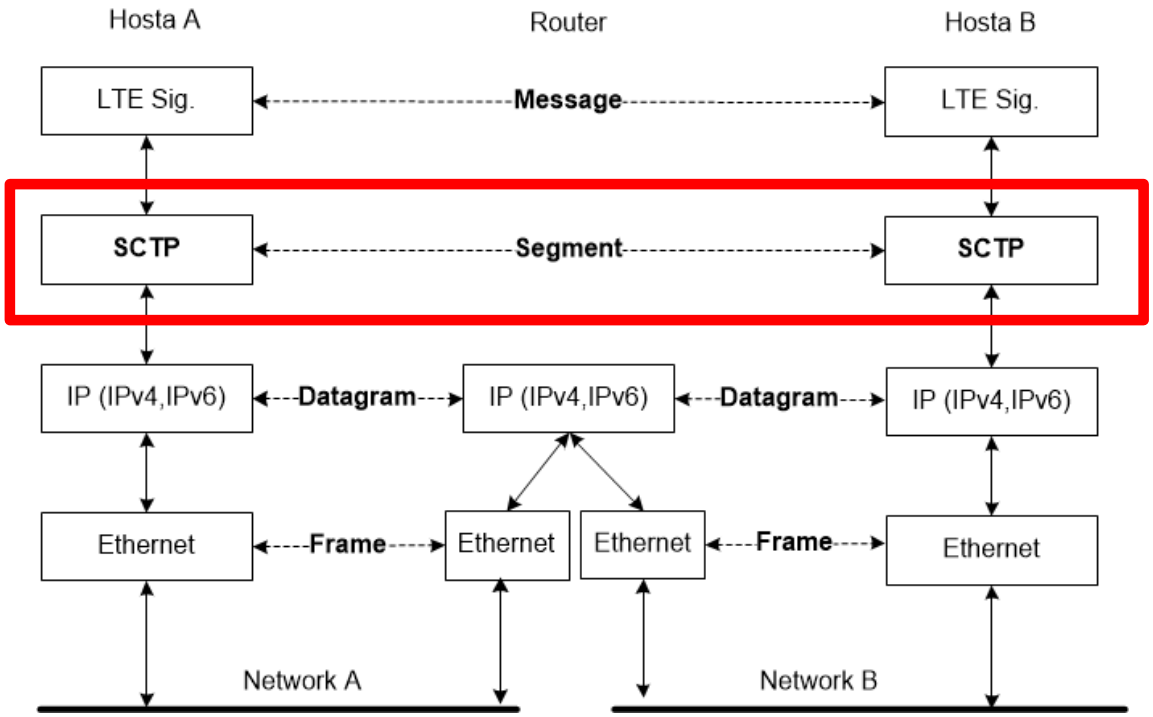


- Dostarcza mechanizmu adresowania aplikacji przez porty
- Zestawiane połączenie
- Zorientowany na przesyłanie strumienia bajtów
- Retransmituje zagubione segmenty i dba o kolejność bajtów
- Wykrywa przeciążenia w sieci i implementuje mechanizm sterowania przepływem
- Obsługuje tylko adresy typu **unicast**

Protokół **SCTP** – sterowanie transmisją dla komunikatów

- Steruje wymianą danych pomiędzy dwoma stacjami - tylko dla adresacji **unicast**
- Zapewnia niezawodność – potwierdzenia, automatyczne ponawianie wysyłania segmentów, szacowanie RTT
- Sesja SCTP – może obejmować kilka adresów dla jednej stacji (jednego końca sesji) - **multihoming**.
- W obrębie jednej sesji dane mogą być przesyłane w niezależnych strumieniach
- SCTP jest zorientowany na komunikaty i zapewnia zachowanie granicy komunikatów po stronie odbiorczej.
- SCTP dostosowuje prędkość wysyłanych danych do warunków w sieci podobnie jak TCP. W tym względzie zapewniona współpraca z TCP (strumienie TCP i SCTP powinny uzyskiwać tą samą przepływność).
- Na jednym gnieździe może łączyć się z wieloma procesami zdalnymi

Warstwa transportowa SCTP w sieci TCP/IP wpływ na funkcjonalność aplikacji



- Dostarcza mechanizmu adresowania aplikacji przez porty
- Zestawiane połączenie, kilka strumieni dla jednego poł.
- Zorientowany na przesyłanie **komunikatów**
- Retransmituje zagubione segmenty i dba o kolejność bajtów
- Wykrywa przeciążenia w sieci i implementuje mechanizm sterowania przepływem
- Obsługuje tylko adresy typu unicast
- Multihoming

PROTOKÓŁ TRANSPORTOWY TCP

Połączenie TCP

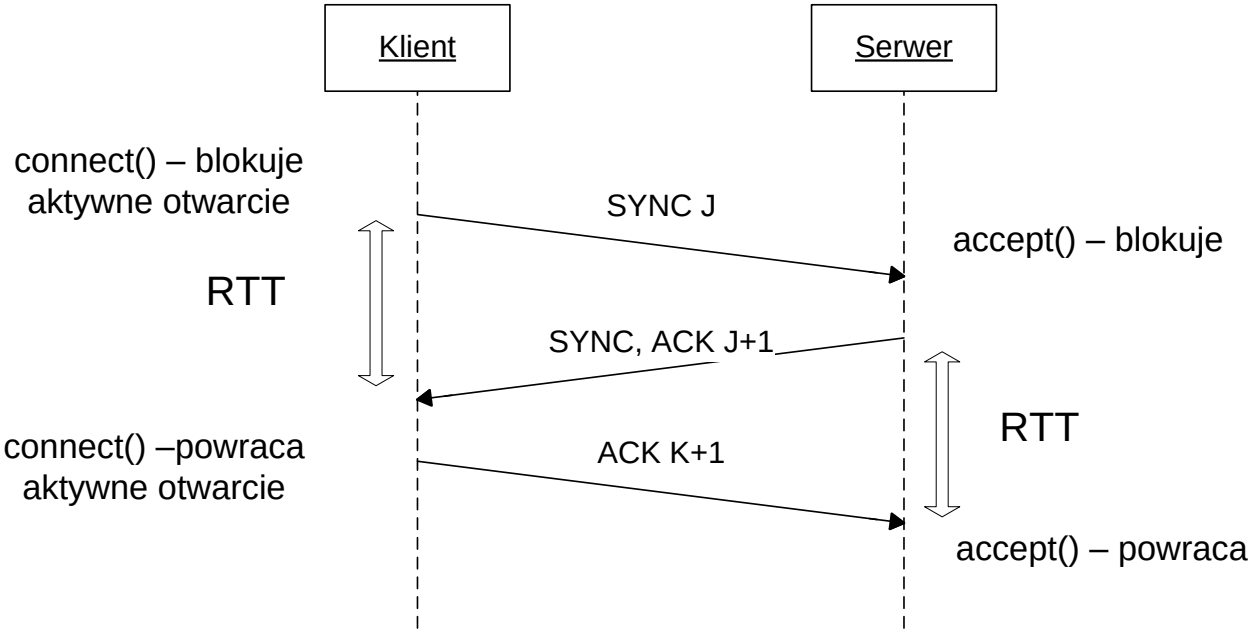
Fazy połączenia (sesji) protokołu TCP:

- Nawiązanie połączenia
- Przekaz danych
- Zakończenie połączenia

Stany protokołu TCP – z każdą fazą połączenia protokołu TCP związane są stany

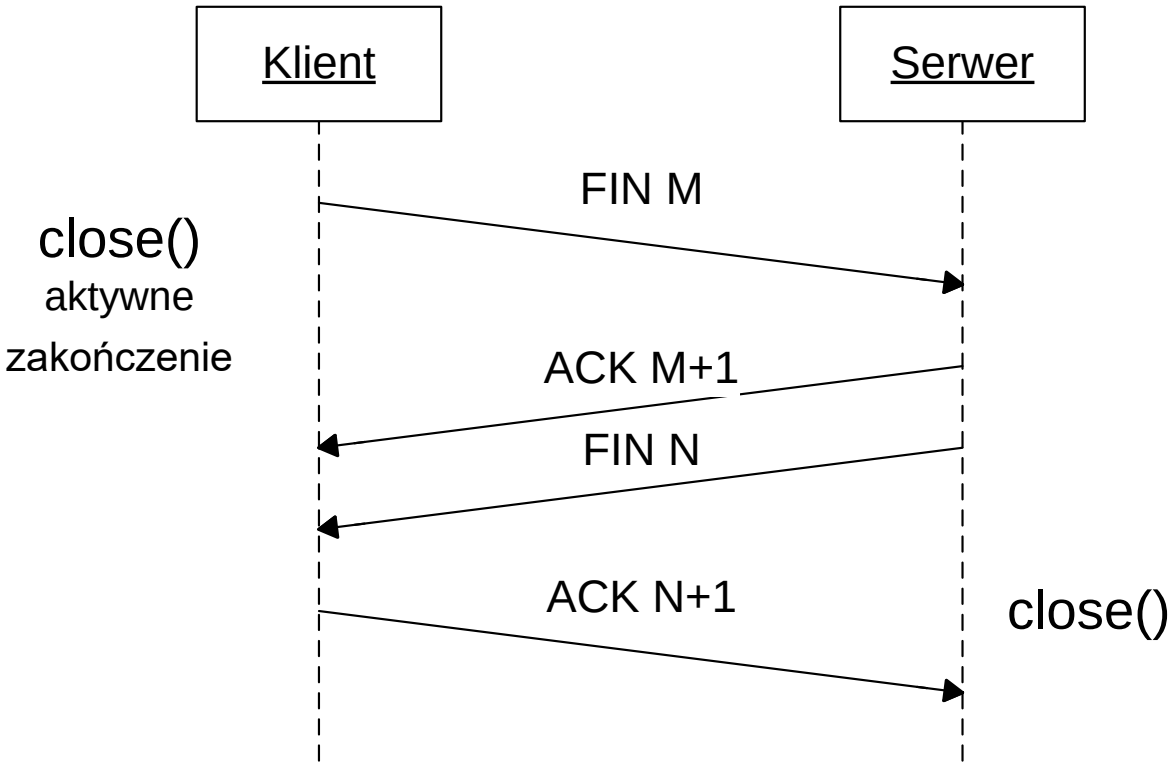
TCP - Nawiązywanie połączenia

- Uzgadnianie trójfazowe (three way handshake)
  - Otwarcie pasywne (Serwer: socket(), bind(), listen())
  - Otwarcie aktywne – Klient: connect() – wysyła SYN J
  - Serwer wysyła potwierdzenie SYN J (ACK J+1) i SYN K
  - Klient wysyła potwierdzenie SYN K (ACK K+1)

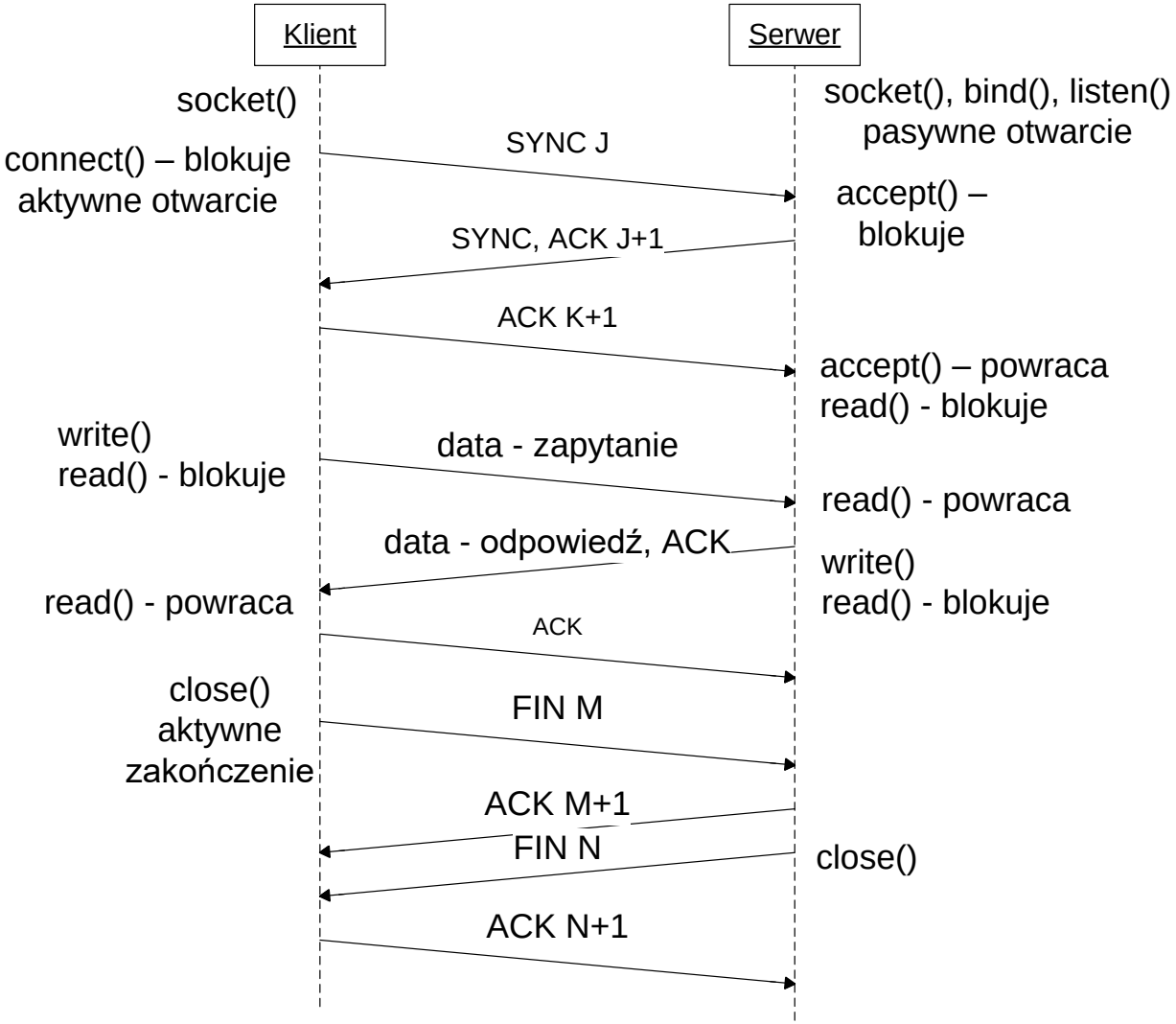


TCP – Zakończenie połączenia

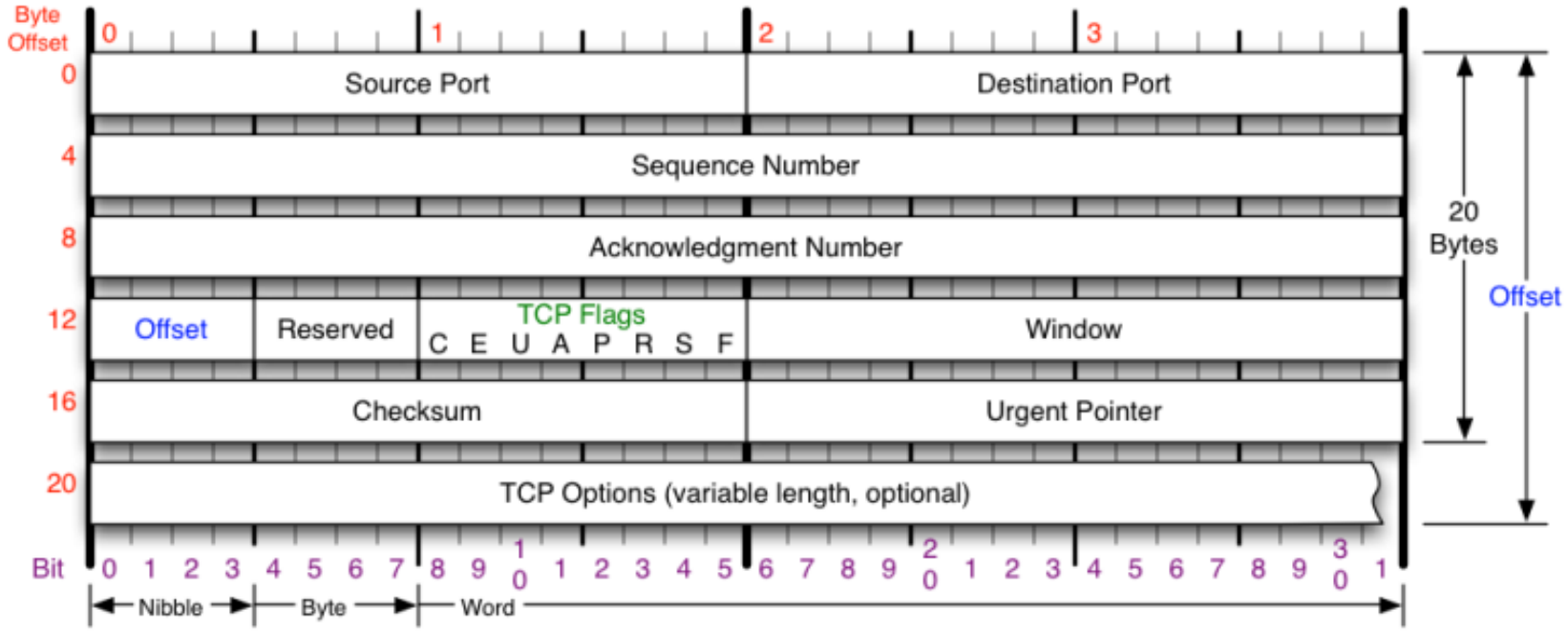
- Pierwszy punkt końcowy (klient lub serwer) wywołuje funkcję close() – Zamknięcie aktywne – wysyłany segment FIN
- Drugi punkt końcowy odbiera FIN, wykonuje zamknięcie bierne, potwierdza FIN
- Drugi punkt końcowy wywołuje close() i TCP wysyła FIN
- Pierwszy punkt końcowy wysyła potwierdzenie



TCP – wymiana pakietów przez połączenie TCP



Nagłówek TCP



TCP Flags

C E U A P R S F

Congestion Window
C 0x80 Reduced (CWR)
E 0x40 ECN Echo (ECE)
U 0x20 Urgent
A 0x10 Ack
P 0x08 Push
R 0x04 Reset
S 0x02 Syn
F 0x01 Fin

Congestion Notification

ECN (Explicit Congestion Notification). See RFC 3168 for full details, valid states below.

Packet State	DSB	ECN bits
Syn	0 0	1 1
Syn-Ack	0 0	0 1
Ack	0 1	0 0
No Congestion	0 1	0 0
No Congestion	1 0	0 0
Congestion	1 1	0 0
Receiver Response	1 1	0 1
Sender Response	1 1	1 1

TCP Options

0 End of Options List
1 No Operation (NOP, Pad)
2 Maximum segment size
3 Window Scale
4 Selective ACK ok
8 Timestamp

Checksum

Checksum of entire TCP segment and pseudo header (parts of IP header)

Offset

Number of 32-bit words in TCP header, minimum value of 5. Multiply by 4 to get byte count.

RFC 793

Please refer to RFC 793 for the complete Transmission Control Protocol (TCP) Specification.

Wymiana pakietów przez połączenie TCP – TCPDUMP/loopback(1/2)

1. 15:05:07.225102 IP6 (hlim 64, next-header TCP (6) payload length: 40) ::1.45670 > ::1.13: Flags [S], cksum 0x0030 (incorrect -> 0x8ab4), seq 505526261, win 43690, options [mss 65476,sackOK,TS val 61770835 ecr 0,nop,wscale 10], length 0
2. 15:05:07.225137 IP6 (hlim 64, next-header TCP (6) payload length: 40) ::1.13 > ::1.45670: Flags [S.], cksum 0x0030 (incorrect -> 0x6128), seq 291276829, ack 505526262, win 43690, options [mss 65476,sackOK,TS val 61770835 ecr 61770835,nop,wscale 10], length 0
3. 15:05:07.225184 IP6 (hlim 64, next-header TCP (6) payload length: 32) ::1.45670 > ::1.13: Flags [.], cksum 0x0028 (incorrect -> 0x3488), ack 1, win 43, options [nop,nop,TS val 61770835 ecr 61770835], length 0
4. 15:05:07.225338 IP6 (hlim 64, next-header TCP (6) payload length: 58) ::1.13 > ::1.45670: Flags [P.], cksum 0x0042 (incorrect -> 0x31bb), seq 1:27, ack 1, win 43, options [nop,nop,TS val 61770835 ecr 61770835], length 26

Wymiana pakietów przez połączenie TCP – TCPDUMP/loopback(2/2)

5. 15:05:07.225385 IP6 (hlim 64, next-header TCP (6) payload length: 32) ::1.13 > ::1.45670: Flags [F.], cksum 0x0028 (incorrect -> 0x346d), seq 27, ack 1, win 43, options [nop,nop,TS val 61770835 ecr 61770835], length 0
6. 15:05:07.225399 IP6 (hlim 64, next-header TCP (6) payload length: 32) ::1.45670 > ::1.13: Flags [.], cksum 0x0028 (incorrect -> 0x346e), ack 27, win 43, options [nop,nop,TS val 61770835 ecr 61770835], length 0
7. 15:05:07.225671 IP6 (hlim 64, next-header TCP (6) payload length: 32) ::1.45670 > ::1.13: Flags [F.], cksum 0x0028 (incorrect -> 0x346c), seq 1, ack 28, win 43, options [nop,nop,TS val 61770835 ecr 61770835], length 0
8. 15:05:07.225710 IP6 (hlim 64, next-header TCP (6) payload length: 32) ::1.13 > ::1.45670: Flags [.], cksum 0x0028 (incorrect -> 0x346c), ack 2, win 43, options [nop,nop,TS val 61770835 ecr 61770835], length 0

Wymiana pakietów przez połączenie TCP – TCPDUMP/Ethernet (1/2)

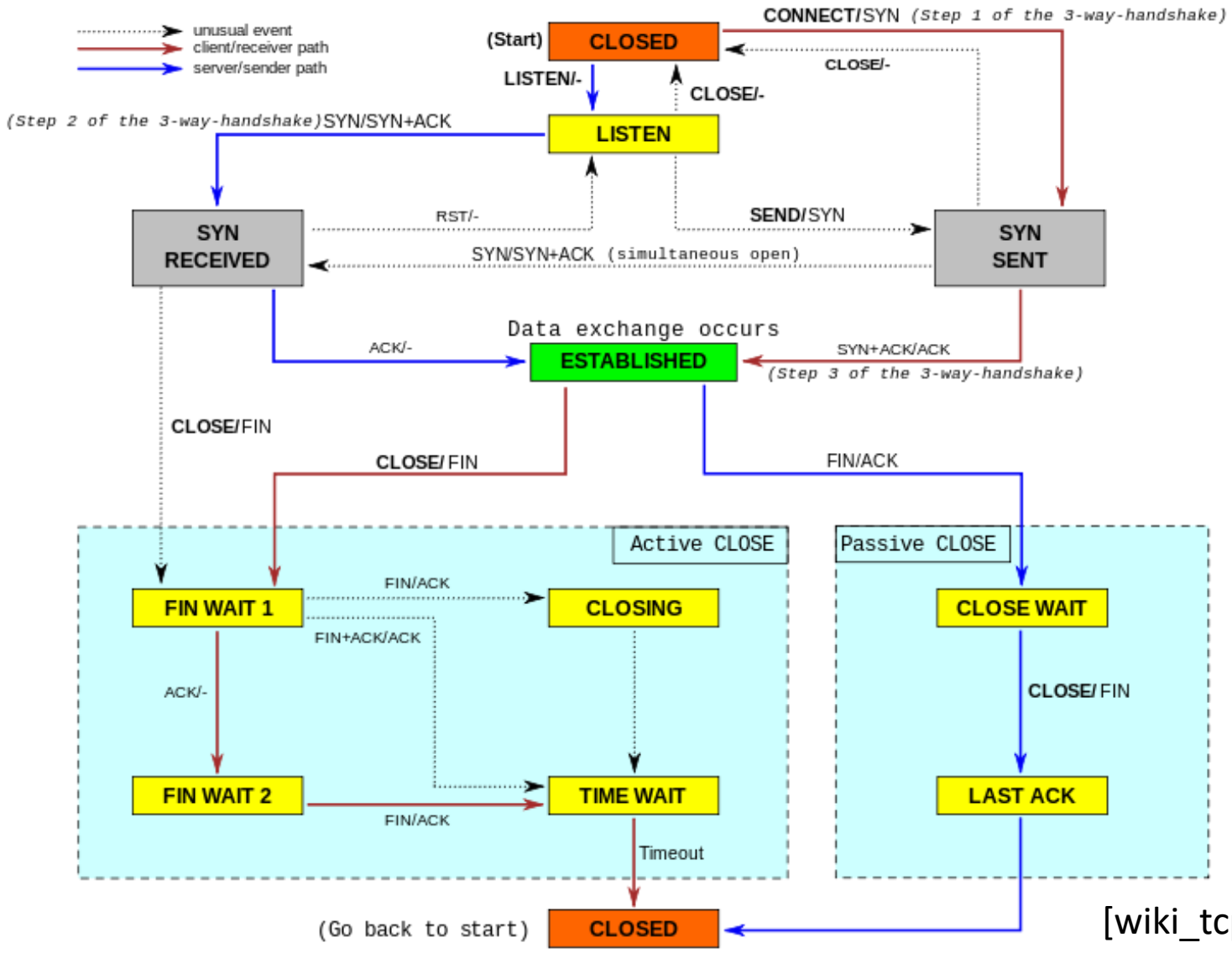
1. 08:55:15.701236 IP6 (hlim 64, next-header TCP (6) payload length: 40) cli.42048 > serv.13: Flags [S], cksum 0x845f (correct), seq 2400824103, win 5760, options [mss 1440,sackOK,TS val 2134988560 ecr 0,nop,wscale 7], length 0
2. 08:55:15.701377 IP6 (hlim 64, next-header TCP (6) payload length: 40) serv.13 > cli.42048: Flags [S.], cksum 0xf939 (correct), seq 1471661134, ack 2400824104, win 28560, options [mss 1440,sackOK,TS val 65475092 ecr 2134988560,nop,wscale 10], length 0
3. 08:55:15.701705 IP6 (hlim 64, next-header TCP (6) payload length: 32) cli.42048 > serv.13: Flags [.], cksum 0x9757 (correct), ack 1, win 45, options [nop,nop,TS val 2134988561 ecr 65475092], length 0
4. 08:55:15.702011 IP6 (hlim 64, next-header TCP (6) payload length: 58) serv.13 > cli.42048: Flags [P.], cksum 0x7f9d (correct), seq 1:27, ack 1, win 28, options [nop,nop,TS val 65475093 ecr 2134988561], length 26

Wymiana pakietów przez połączenie TCP – TCPDUMP/Ethernet (1/2)

5. 08:55:15.702050 IP6 (hlim 64, next-header TCP (6) payload length: 32) serv.13 > cli.42048: Flags [F.], cksum 0x974c (correct), seq 27, ack 1, win 28, options [nop,nop,TS val 65475093 ecr 2134988561], length 0
6. 08:55:15.702326 IP6 (hlim 64, next-header TCP (6) payload length: 32) serv.42048 > cli.13: Flags [.], cksum 0x973b (correct), ack 27, win 45, options [nop,nop,TS val 2134988562 ecr 65475093], length 0
7. 08:55:15.702391 IP6 (hlim 64, next-header TCP (6) payload length: 32) cli.42048 > serv.13: Flags [F.], cksum 0x9739 (correct), seq 1, ack 28, win 45, options [nop,nop,TS val 2134988562 ecr 65475093], length 0
8. 08:55:15.702417 IP6 (hlim 64, next-header TCP (6) payload length: 32) serv.13 > cli.42048: Flags [.], cksum 0x974a (correct), ack 2, win 28, options [nop,nop,TS val 65475093 ecr 2134988562], length 0

TCP – Diagram przejść

http://commons.wikimedia.org/wiki/File:Tcp_state_diagram_fixed.svg



[wiki_tcp]

TCP: Stan TIME_WAIT

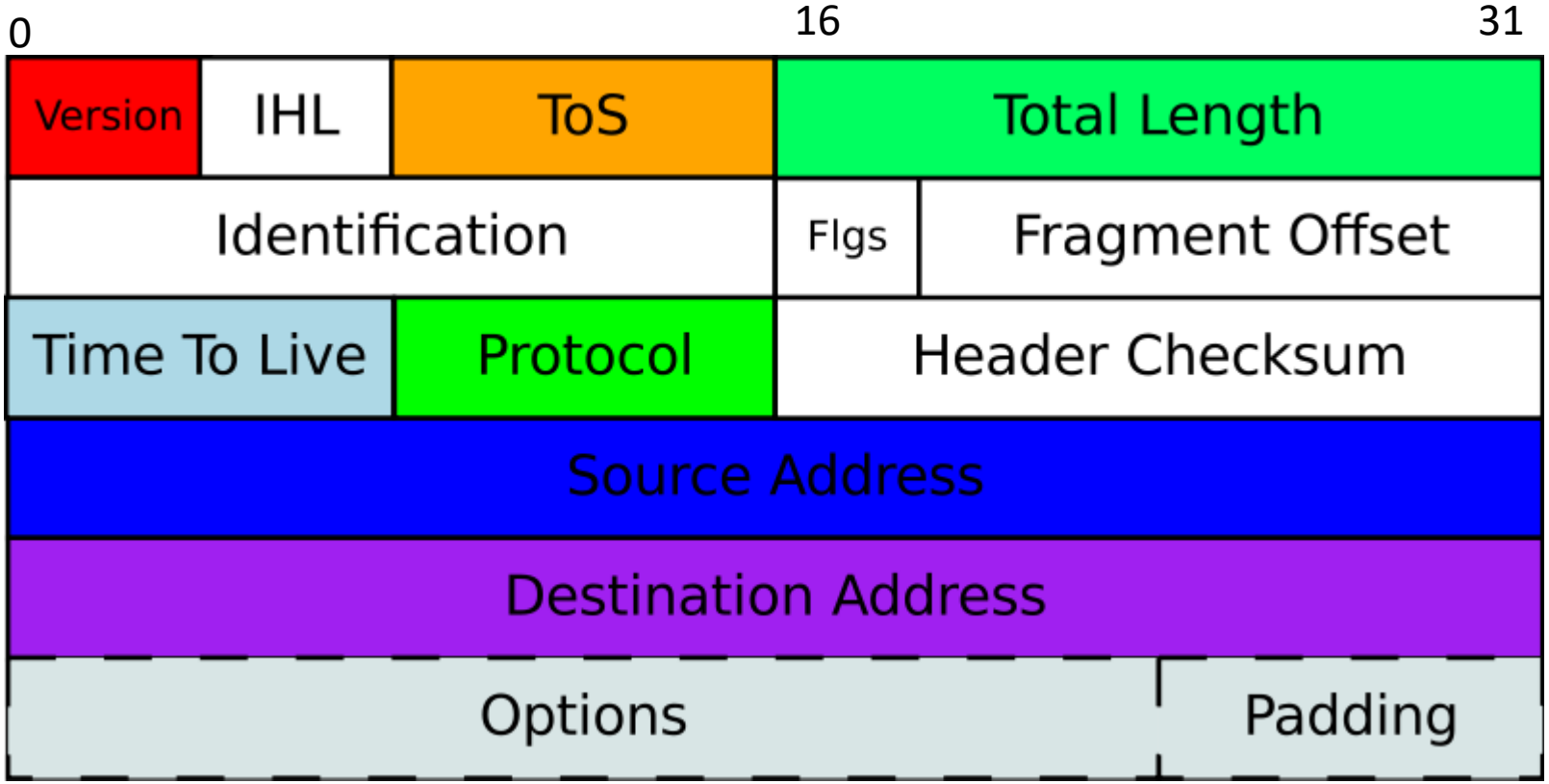
- Punkt końcowy, który wykonuje aktywne zakończenie przechodzi przez stan **TIME_WAIT**
- Stan **TIME_WAIT** trwa 2xMSL (*Maximum Segment Lifetime*)
- MSL = 30s lub 2 min
- Czas trwania TIME_WAIT – od 1 do 4 min
- Służy do poprawnego zakończenia połączenia i zminimalizowania prawdopodobieństwa, że nowe połączenie otrzyma zagubione datagramy z wcześniejszego połączenia

Rozmiary buforów i ograniczenia Ustawianie parametrów połączeń

getsockopt()/setsockopt()

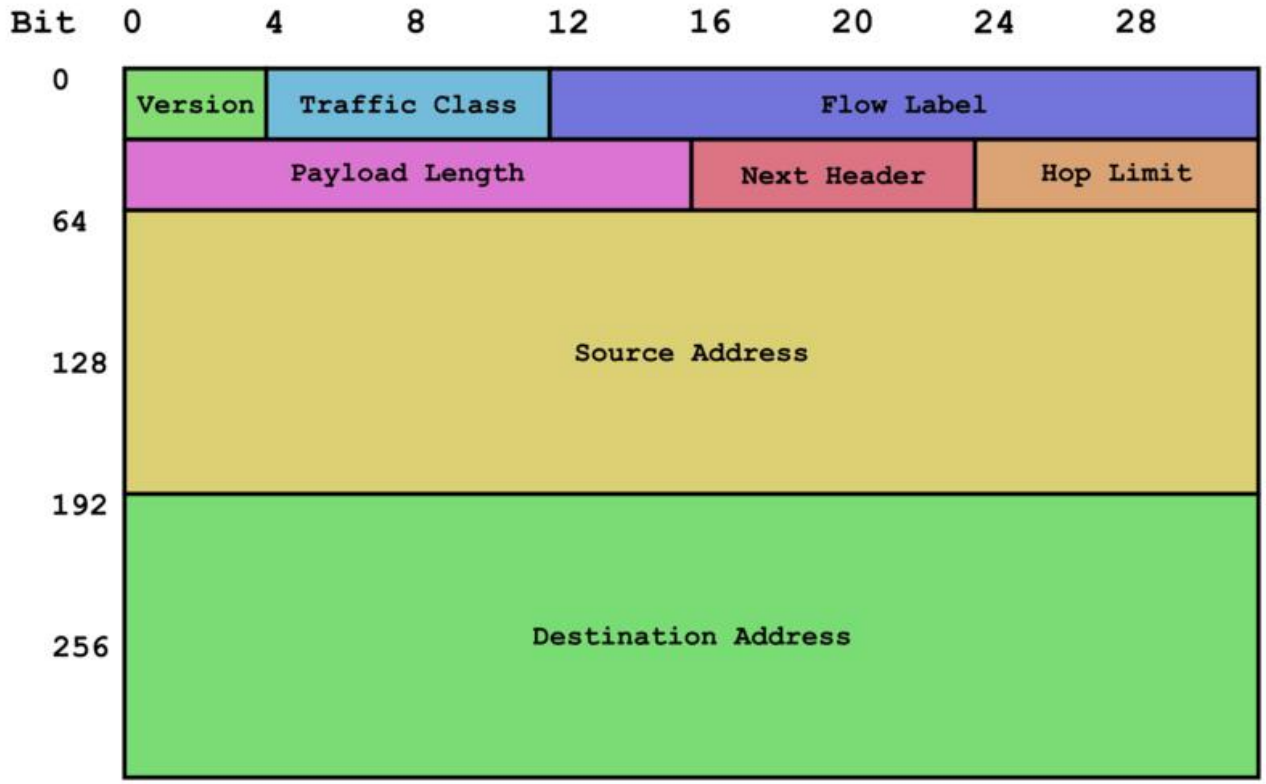
Rozmiary buforów i ograniczenia(1)

- Max. Rozmiar datagramu IPv4 wynosi 65535 B (pole Total Length w nagłówku):



Rozmiary buforów i ograniczenia (2)

- Max. Rozmiar datagramu IPv6 wynosi 65575 B (pole Payload Length w nagłówku z możliwością rozszerzenia do 32 bitów przez dodatkowe opcje):



Rozmiary buforów i ograniczenia (3)

- Min. rozmiar MTU (***Maximum Transfer Unit***) dla protokołu IPv4 wynosi 68 B (nagłówek + opcje + dane)
- Min. rozmiar MTU dla protokołu IPv6 wynosi 1280 B (RFC 2460 [Deering and Hinden 1998]).
- Najmniejsza wartość MTU na ścieżce między dwiema stacjami nazywa się jednostką MTU dla ścieżki (***Path MTU***)
- Jeśli rozmiar datagramu jest większy niż jednostka MTU dla danego łącza następuje fragmentacja datagramu:
  - Dla IPv4 : stacje końcowe i rutery
  - Dla IPv6 tylko stacje końcowe
  - Ustawiony bit DF w nagłówku IPv4 zabrania fragmentacji pakietu w sieci
- Minimalny gwarantowany rozmiar bufora odbiorczego:
  - IPv4 - **576** B
  - IPv6 – **1500** B

Rozmiary buforów i ograniczenia (4)

- TCP MSS (*Maximum Segment Size*) – maksymalny rozmiar danych jaki wysyła TCP – ogłaszany podczas ustanawiania połączenia:
  - Zwykle: MSS = MTU – [Rozmiar nagłówka IP+TCP]
  - Nie powinno się dopuszczać do fragmentacji

Opcje gniazd

- Sposoby pobierania i ustanawiania opcji gniazd:
  - Funkcje **getsockopt()** i **setsockopt()**
  - Funkcja fcntl()
  - Funkcja ioctl()
- Główne typy (poziomy) opcji:
  - Dla gniazd
  - Dla warstwy sieciowej: IPv4 i IPv6
  - Dla protokołów transportowych: TCP, UDP, SCTP

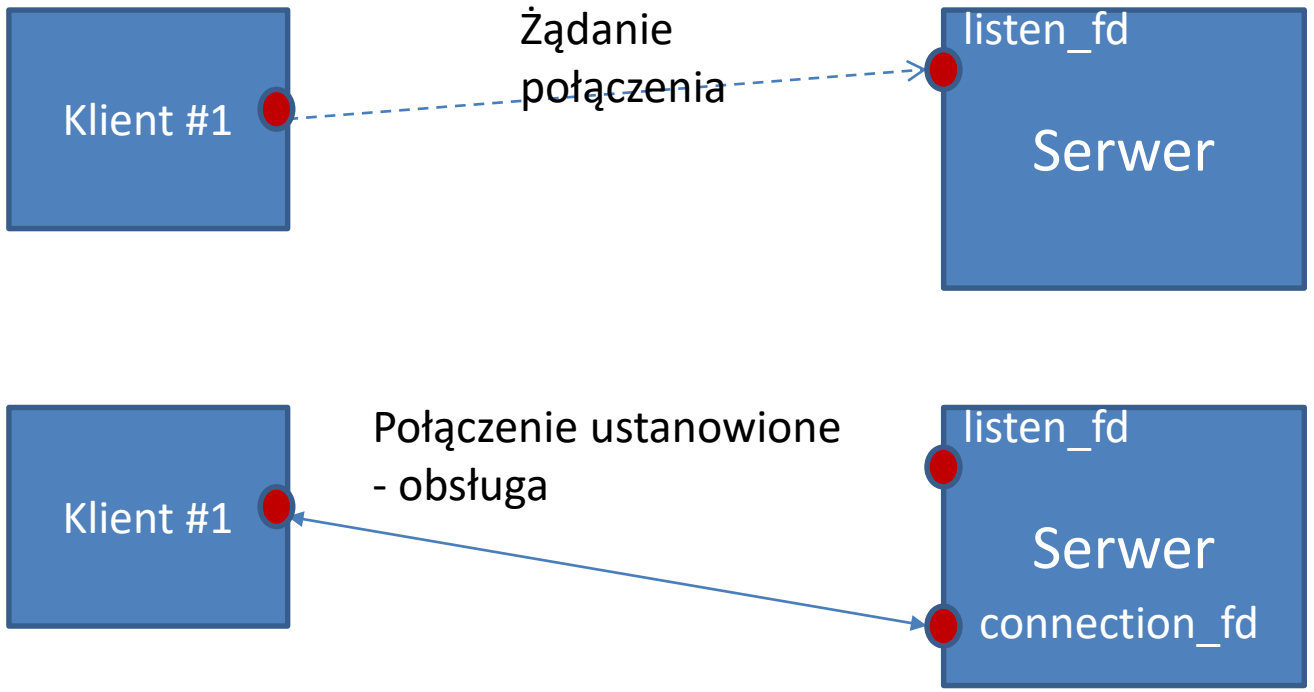
Funkcje getsockopt() i setsockopt()

```
#include <sys/types.h>
#include <sys/socket.h>


int getsockopt(int sockfd, int level, int optname,
               void *optval, socklen_t *optlen);


int setsockopt(int sockfd, int level, int optname,
               const void *optval, socklen_t optlen);
```

Dziedziczenie opcji gniazd dla TCP

- Opcje dziedziczone przez gniazdo połączone z gniazda nasłuchującego:
  - **SO_DEBUG, SO_DONTROUTE, SO_KEEPALIVE, SO_LINGER, SO_OOBINLINE, SO_RCVBUF, SO_RCVLOWAT, SO_SNDBUF, SO_SNDLOWAT, TCP_MAXSEG i TCP_NODELAY**.
- Dla TCP powyższe opcje muszą być ustawione na gnieździe nasłuchującym aby mogły być ustawione na gnieździe połączonym

Dziedziczenie opcji gniazd



OPCJE do ustawiania rozmiarów buforów na poziomie SOL_SOCKET

- **SO_RCVBUF** Ustawia lub pobiera maksymalny rozmiar bufora odbiorczego. Przy ustawianiu (setsockopt()) jądro **podwaja** wartość ustawianą (wykorzystuje dodatkową przestrzeń do przechowywania struktur danych jądra). Podwójna wartość tej opcji jest zwracana przez funkcję getsockopt(). Domyślna wartość tej opcji jest ustawiana z pliku ***/proc/sys/net/core/rmem_default***, a maksymalna wartość tej opcji jest ograniczona przez wartość ustawioną w pliku ***/proc/sys/net/core/rmem_max*** file. Minimalna wartość tej opcji wynosi 256. Dla protokołu TCP (zarówno dla IPv4 jak IPv6) obowiązują wartości z pliku **/proc/sys/net/ipv4/tcp_rmem**.
- **SO_SNDBUF** - Ustawia lub pobiera maksymalny rozmiar bufora nadawczego. Przy ustawianiu (setsockopt()) jądro **podwaja** wartość ustawianą (wykorzystuje dodatkową przestrzeń do przechowywania struktur danych jądra). Podwójna wartość tej opcji jest zwracana przez funkcję getsockopt(). Domyślna wartość tej opcji jest ustawiana z pliku ***/proc/sys/net/core/wmem_default,*** a maksymalna wartość tej opcji jest ograniczona przez wartość ustawioną w pliku ***/proc/sys/net/core/wmem_max***. Minimalna wartość jej opcji wynosi 2048. Dla protokołu TCP (zarówno dla IPv4 jak IPv6) obowiązują wartości z pliku **/proc/sys/net/ipv4/tcp_wmem**
- **TCP_MAXSEG** - *Maksymalny rozmiar segmentu TCP. Nie może być większe niż MTU (Maximal Transmission Unit) interfejsu, przez który będą wysłane segmenty.*

Zmiana domyślnych parametrów w systemie plików **/proc**

- echo ‘wartość’ > /proc/sys/’scieżka_ do_parametru’
- Komenda **sysctl (/proc/sys):**
  - **sysctl -n kernel.hostname**
  - **sysctl -n kernel.hostname=CENTOS**
  - **sysctl net/core/rmem_max**
- Plik konfiguracyjny:
  - /etc/sysctl.conf
    net.ipv4.ip_forward = 0

Przykład pobierania i ustawiania danej opcji

```
1.  int len, rcvbuf;
2.  rcvbuf = 9000;
3.  if( setsockopt(sockfd, SOL_SOCKET, SO_RCVBUF, &rcvbuf,
    sizeof(rcvbuf)) == -1){
4.      fprintf(stderr,"setsockopt error : %s\n",
        strerror(errno));
5.      return 1;
6.  }
7.  len = sizeof(rcvbuf);
8.  if( getsockopt(sockfd, SOL_SOCKET, SO_RCVBUF, &rcvbuf,
    &len) == -1){
9.      fprintf(stderr,"getsockopt error : %s\n",
        strerror(errno));
10.     return 4;
11. }
12. printf("SO_RCVBUF = %d (after setting it to 9000)\n",
    rcvbuf);
```

Najważniejsze OPCJE – SOL_SOCKET

- **SO_RCVLOWAT, SO_SNDLOWAT** – znaczniki dla funkcji select() i pool() dolnego ograniczenia dla buforów odbiorczego i nadawczego
- **SO_KEEPALIVE** - Wysyłanie tzw. pakietów keep-alive wł/wył (0/1). Pakiety te pełnią rolę diagnostyczną - badają, czy zdalna maszyna jest w stanie obsługiwać nasze połączenie. Tylko gniazda połączeniowe. Trzeba pamiętać, że standardowo odstęp czasowy między tymi pakietami wynosi 2h (7200s).
- **SO_OOBINLINE** - Jeśli opcja włączona, to dane typu out-of-band będą wymieszane razem z danymi właściwymi w strumieniu wejściowym. Oznacza to, ze dane OOB będą odbierane przy każdym wywołaniu **recv()**. Jeśli opcja ta będzie wyłączona (domyślnie) to OOB będzie odbierane tylko w przypadku dołączenia flagi MSG_OOB w wywołaniu **recv()**.

Najważniejsze OPCJE – SOL_SOCKET

- **SO_BINDTODEVICE** - Związuje gniazdo z konkretnym interfejsem sieciowym. Wartość dla tej opcji to nazwa interfejsu (np. *eth0*). Takie gniazdo będzie otrzymywało tylko pakiety odebrane poprzez wskazany interfejs.
- **SO_REUSEADDR** - Normalnie funkcja **bind()** zwraca błąd jeśli spróbujemy związać gniazdo z adresem, który jest aktualnie w użyciu. Załóżmy, że gniazdo o adresie 127.0.0.1 i porcie 1111 jest aktualnie w stanie **TCP_FIN**. Jeśli chcielibyśmy z tym adresem skojarzyć jakieś inne gniazdo nie czekając na zakończenie czasochłonnego procesu zamykania połączenia to musimy włączyć właśnie tą opcję. Ważna uwaga: nawet **SO_REUSEADDR** nie pomoże jeśli jakieś gniazdo nasłuchuje (**LISTEN**) na danym adresie.

Najważniejsze OPCJE – SOL_SOCKET

- **SO_TYPE** - Zwraca typ gniazda (SOCK_STREAM, SOCK_DGRAM itp.). Tylko **getsockopt()**.
- **SO_BROADCAST** - Używane tylko dla gniazd datagramowych. Jeśli włączone to gniazdo może odbierać i wysyłać datagramy typu broadcast (do grupy adresów).
- **SO_ERROR** - Pobiera liczbowy kod ostatniego błędu, który wystąpił na danym gnieździe. Tylko **getsockopt()**.

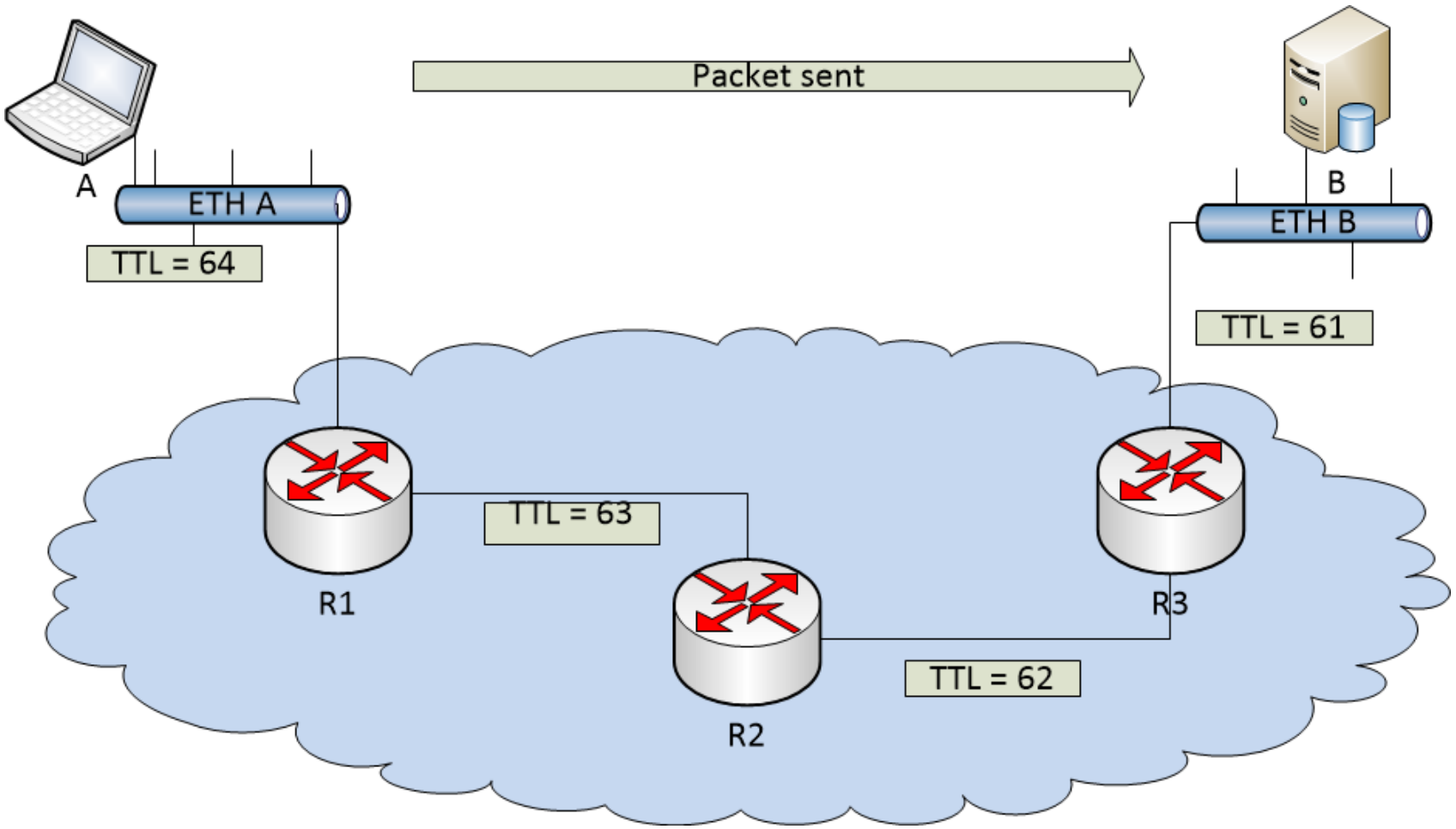
Najważniejsze OPCJE – SOL_SOCKET

- **SO_PRIORITY** - Set the protocol-defined priority for all packets to be sent on this socket. Linux uses this value to order the networking queues: packets with a higher priority may be processed first depending on the selected device queueing discipline. For IP this also sets the IP type-of-service (TOS) field for outgoing packets. Setting a priority outside the range 0 to 6 requires the CAP_NET_ADMIN capability.

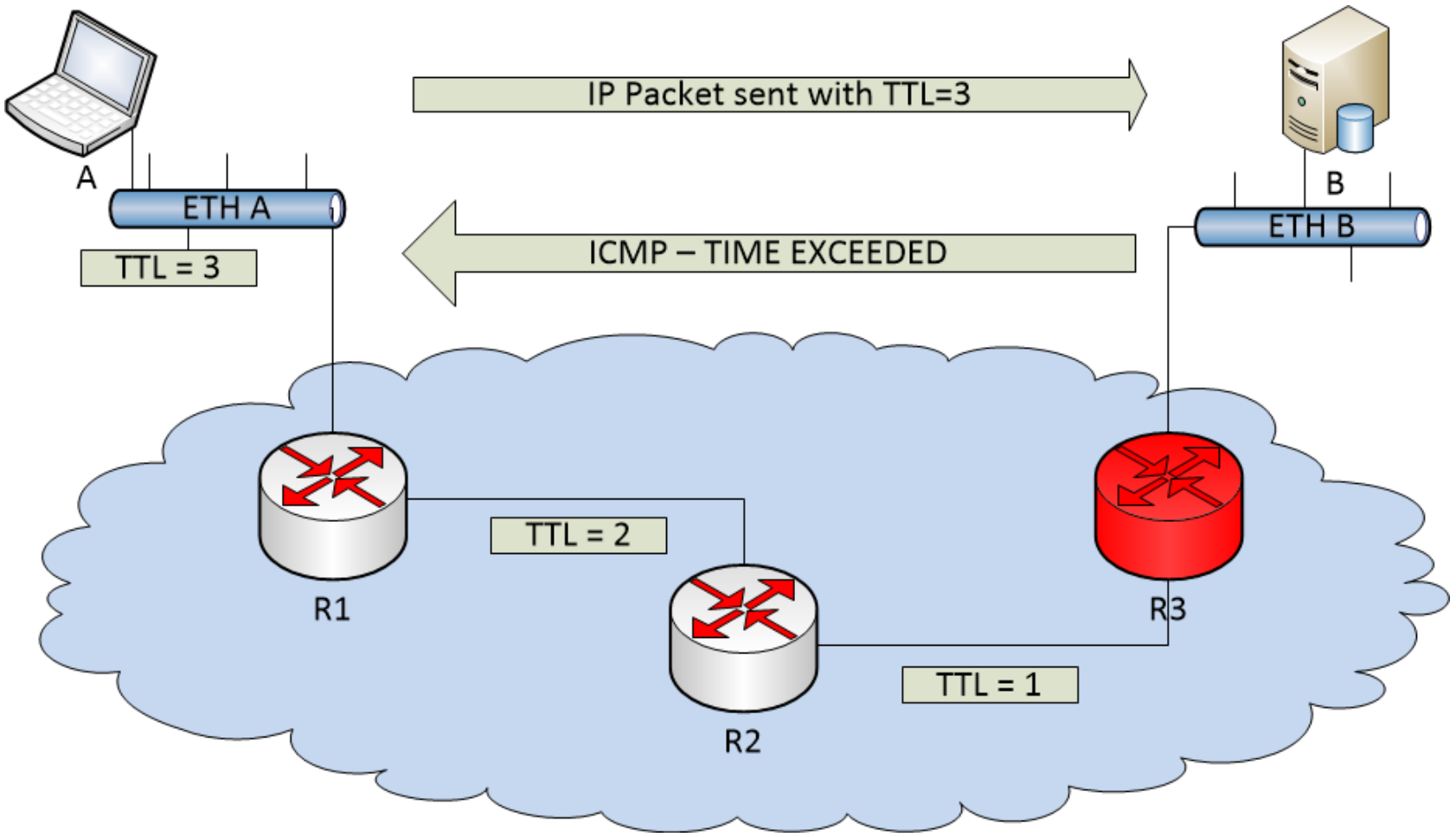
Najważniejsze OPCJE - SOL_IP

- **SOL_IP** - gniazda w domenie PF_INET (*bits/in.h*): IP_OPTIONS - Ustawia/pobiera opcje protokołu IP używane podczas wysyłania pakietów obsługiwanych przez tenże protokół. Znaczenia tej i trzech następnych opcji należy szukać w RFC definiującym protokół IP (RFC791)
- **IP_TOS** - Ustawia/pobiera pole TOS (Type Of Service) w nagłówku IP.
- **IP_MTU** - Pobiera aktualną wartość MTU używaną przez dane gniazdo.
- **IP_TTL** - Ustawia/pobiera pole TTL (Time To Live) nagłówka IP.

IP_TTL (IPV6_UNICAST_HOPS) (1/2)



IP_TTL (IPV6_UNICAST_HOPS) (2/2)



Najważniejsze OPCJE – SOL_IP

- **IP_HDRINCL** - Jeśli włączone to użytkownik podczas wysyłania danych poprzez gniazdo typu SOCK_RAW musi sam konstruować nagłówki IP.
- **IP_ADD_MEMBERSHIP** - Dołącza gniazdo do tzw. grupy multicast. Linux pozwala na wysyłanie multicastów tylko na gniazdach SOCK_DGRAM i SOCK_RAW.
- **IP_DROP_MEMBERSHIP** - Powoduje opuszczenie danej grupy multicast. Argumentem jest ta sama struktura, co w IP_ADD_MEMBERSHIP.

Najważniejsze OPCJE – SOL_IPV6

- **IPV6_UNICAST_HOPS** – analogiczna opcja do IP_TTL
- **IPV6_ADD_MEMBERSHIP**, **IPV6_DROP_MEMBERSHIP** – rozgłaszanie grupowe
- **IPV6_MTU** – pobieranie/ustawianie MTU
- **IPV6_MTU_DISCOVER** – sterowanie wykrywaniem MTU na ścieżce

Najważniejsze OPCJE – SOL_IPV6

- **IPV6_RECVPKTINFO** (od Linux 2.6.14) Ustawia możliwość odbierania informacji sterujących typu IPV6_PKTINFO dla przychodzących datagramów. Format informacji sterujących jest zdefiniowany przez strukturę in6_pktinfo (RFC 3542). Opcja dostępna tylko dla gniazd SOCK_DGRAM lub SOCK_RAW.
- **IPV6_RTHDR, IPV6_AUTHHDR, IPV6_DSTOPTS, IPV6_HOPOPTS, IPV6_FLOWINFO, IPV6_HOPLIMIT** – powyższe opcje umożliwiają uzyskanie informacji zawartych w nagłówkach rozszerzeń z otrzymywanych pakietów. Opcja dostępna tylko dla gniazd SOCK_DGRAM lub SOCK_RAW.

Najważniejsze OPCJE – SOL_IPV6

- **IPV6_TCLASS** – umożliwia ustawienie pola Traffic Class w nagłówku IPv6 wysyłanych pakietów
- **IPV6_RECVTCLASS** – umożliwia pobieranie pola Traffic Class z nagłówka IPv6 przez funkcję recvmsg()
- **IPV6_V6ONLY - (/proc/sys/net/ipv6/ bindv6only)** – umożliwia odbieranie i wysyłanie pakietów IPv4. Jeśli ustawiona na FALSE proces blokuje port zarówno dla protokołu IPv4 jak i IPv6.

Najważniejsze OPCJE - **SOL_TCP**(1/2)

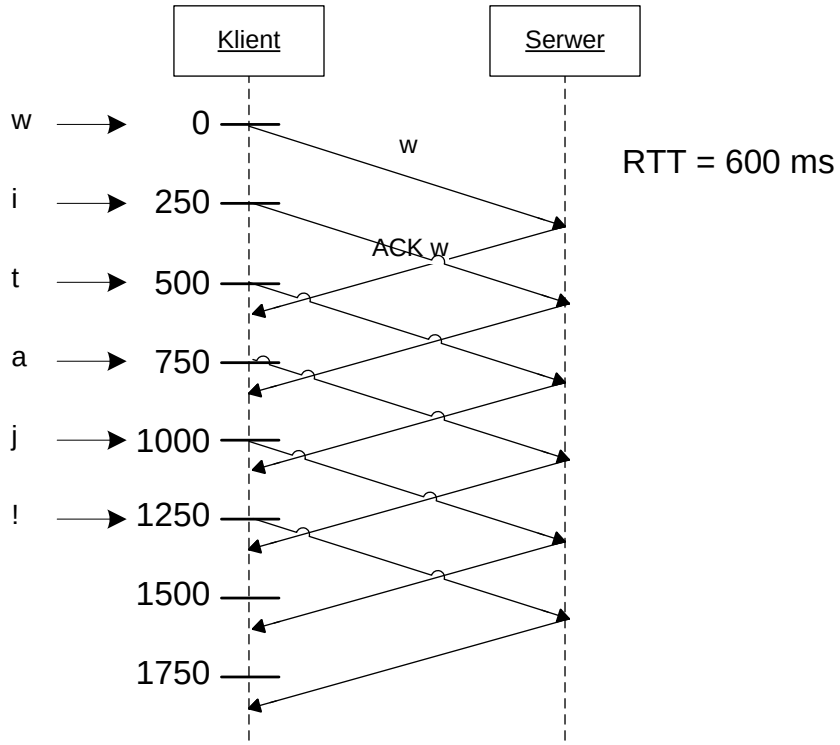
- **SOL_TCP** (domena PF_INET/PF_INET6, typ SOCK_STREAM) (*linux/socket.h*):
- **TCP_MAXSEG** - Maksymalny rozmiar segmentu TCP. Nie może być większe niż MTU (Maximal Transmission Unit) interfejsu, przez który będą wysłane segmenty.
- **TCP_CORK** - Jeśli włączone to wysyłanie wszystkich fragmentów znajdujących się w kolejce zostanie wstrzymane do momentu wyłączenia tej opcji (2.2.x).
- **TCP_KEEPALIVE** – działa tylko jeśli włączona opcja SO_KEEPALIVE

  /proc/sys/net/ipv4/tcp_keepalive_time (default 7200)
  /proc/sys/net/ipv4/tcp_keepalive_intvl (default 75)
  /proc/sys/net/ipv4/tcp_keepalive_probes (default 9)

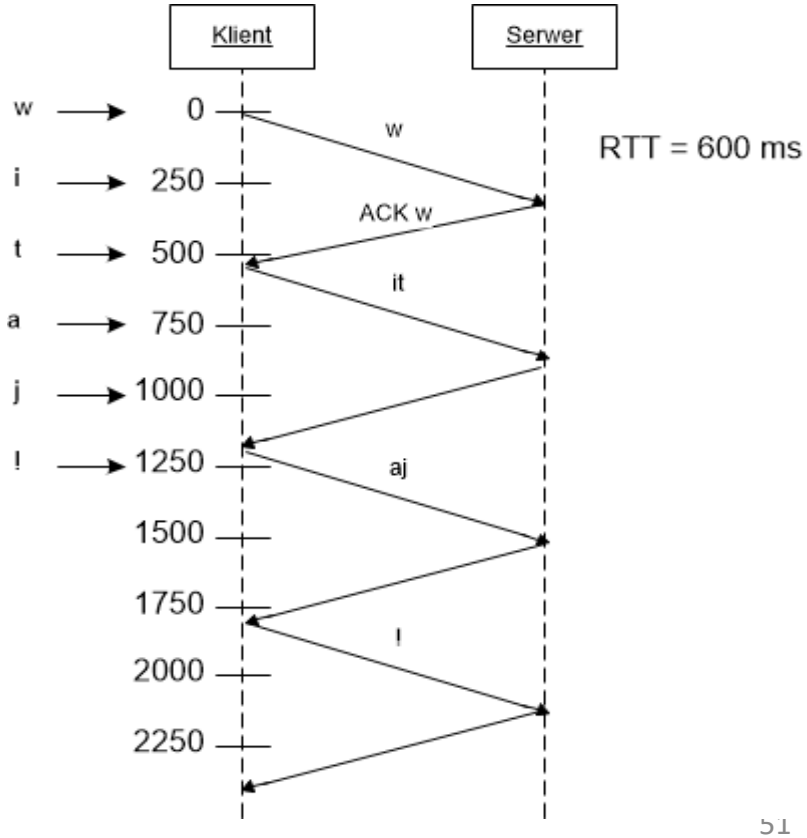
Najważniejsze OPCJE - **SOL_TCP**(2/2)

- **TCP_NODELAY** - Jeśli równe 1 to zostaje wyłączony tzw. algorytm Nagle. Algorytm ten wstrzymuje transmisję segmentów TCP dopóki nie zostanie uzbierana wystarczająca paczka danych. Szczegóły w RFC1122. Włączenie tej opcji jest uzasadnione jeśli zależy nam na wysyłaniu małych porcji danych bez zbędnych opóźnień.

Algorytm Nagla wyłączony



Algorytm Nagla włączony



Opcja **SO_LINGER**

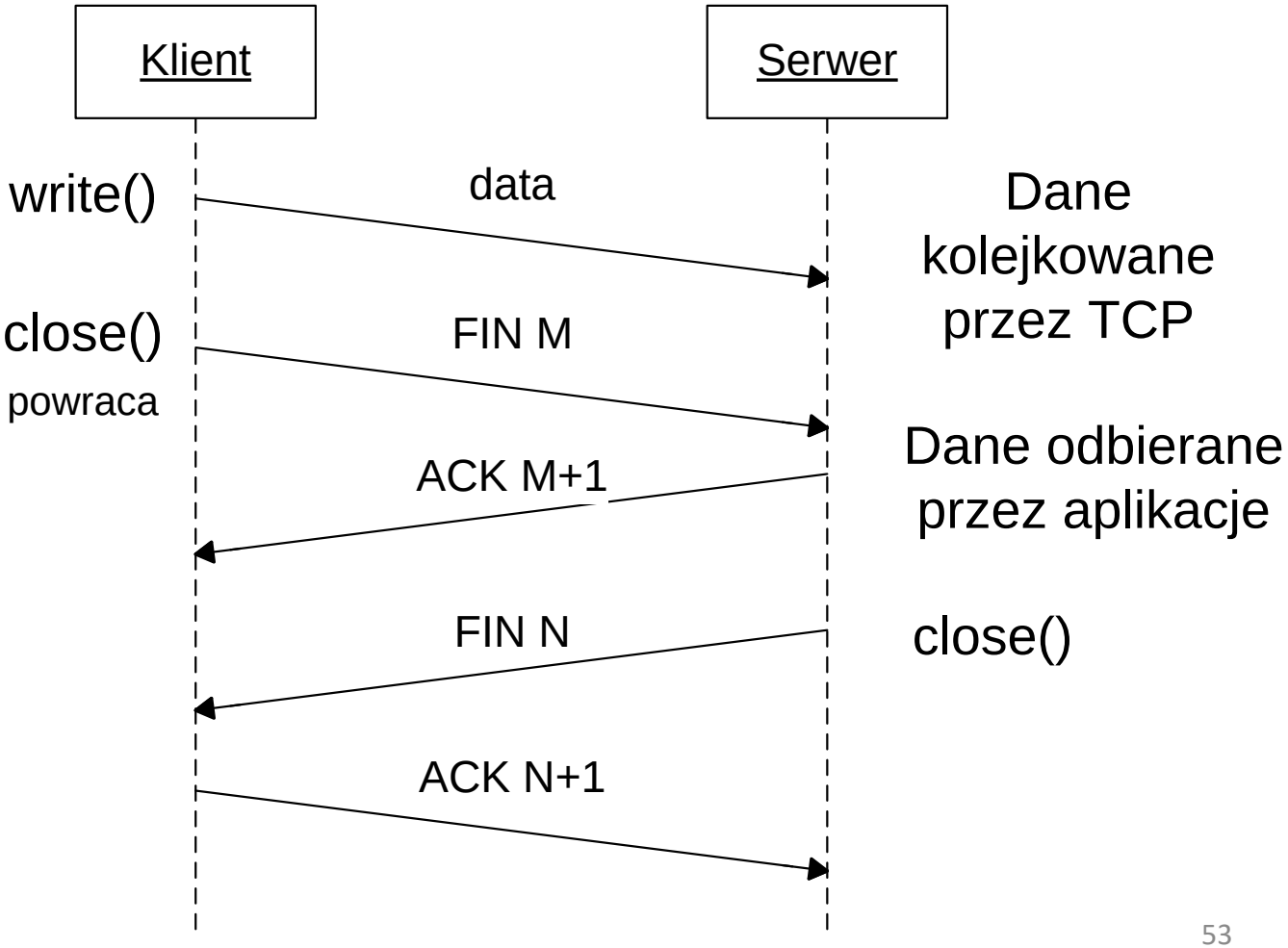
- **SO_LINGER** - Steruje zachowaniem funkcji **close() dla TCP**. Argumentem tej opcji jest struktura:

```
struct linger {
    int l_onoff; /* opcja SO_LINGER wł/wył */
    int l_linger; /* limit czasowy na wysłanie danych */
};
```

  - Jeśli **l_onoff** jest równe 0 to wspomniane funkcje działają standardowo tzn. natychmiast zwracają sterowanie, a wszystkie pakiety znajdujące się w kolejce do wysłania są obsługiwane niezależnie przez jądro.
  - Jeśli natomiast **l_onoff** jest różne od zera to wywołania **close()** oraz **shutdown()** :
    - jeśli **l_linger** jest różne od zera to proces będzie uśpiony dopóki zostaną wysłane wszystkie pakiety przy czym **l_linger** określa maksymalny czas uśpienia (w tym przypadku celowe jest sprawdzanie kodu powrotu z funkcji **close()**)
    - jeśli **l_linger** jest równe zero to wszystkie pakiet znajdujące się w kolejce wysyłkowej zostaną zniszczone oraz nastąpi natychmiastowe zerwanie połączenia poprzez wysłanie segmentu **RST** (**gniazdo nie przechodzi przez stan TIME_WAIT, tylko przechodzi do stanu CLOSED)**.

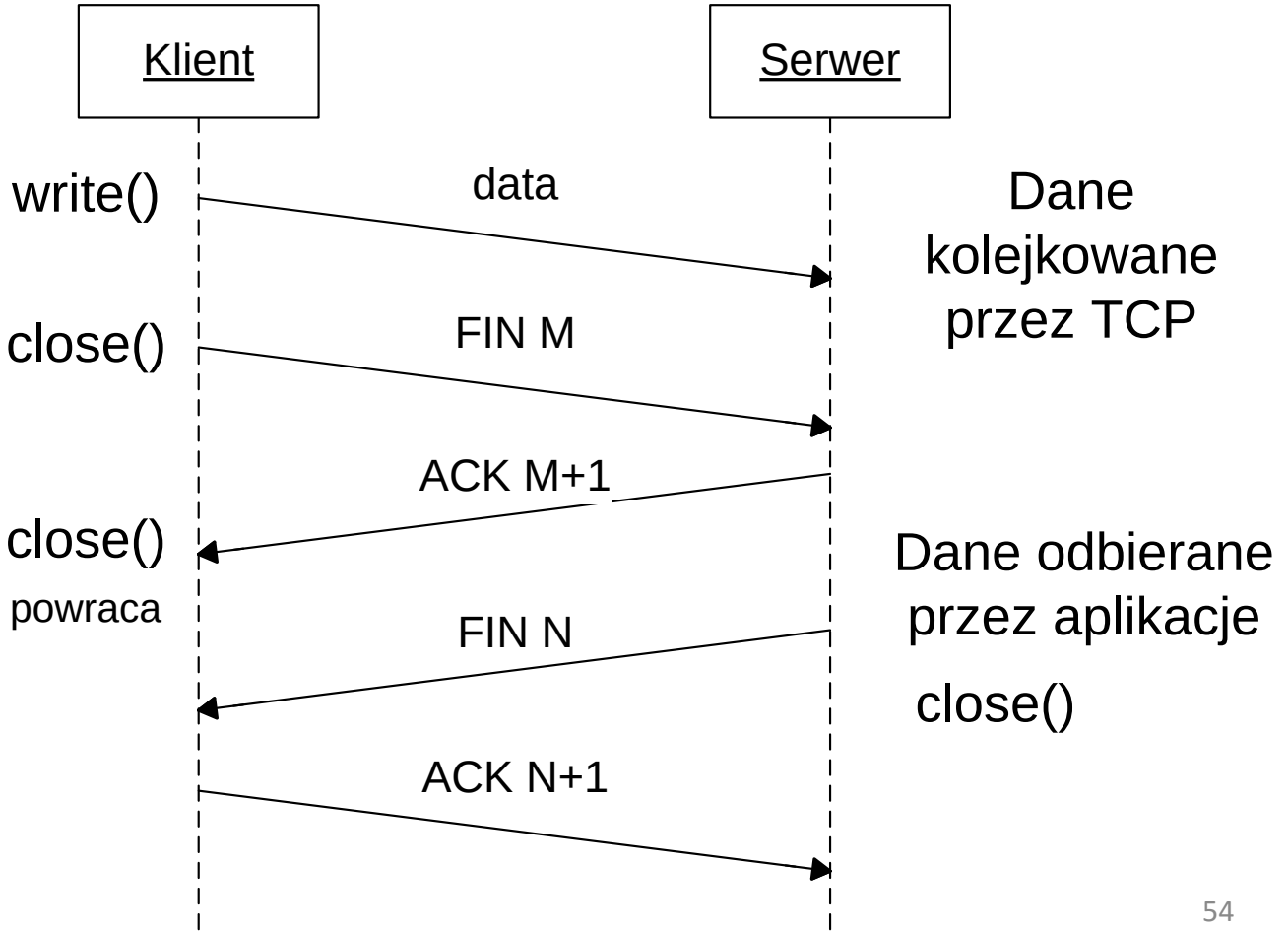
Opcja SO_LINGER nie ustawiona - normalne zakończenie połączenia TCP

```
l_onoff = 0;
```



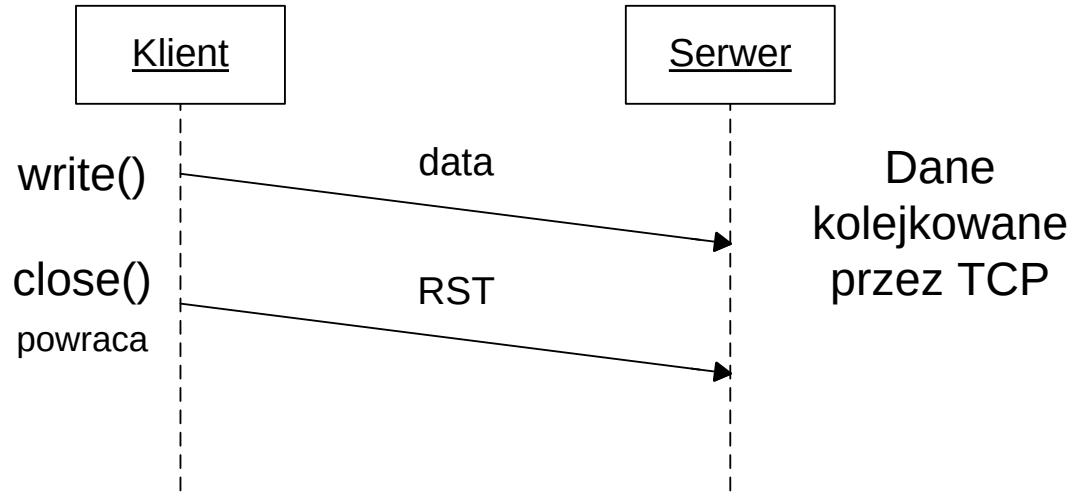
Opcja SO_LINGER ustawiona - zakończenie połączenia TCP z informacją, że druga strona potwierdziła otrzymanie segmentu FIN

```
l_onoff =1;
l_linger > 0;
```

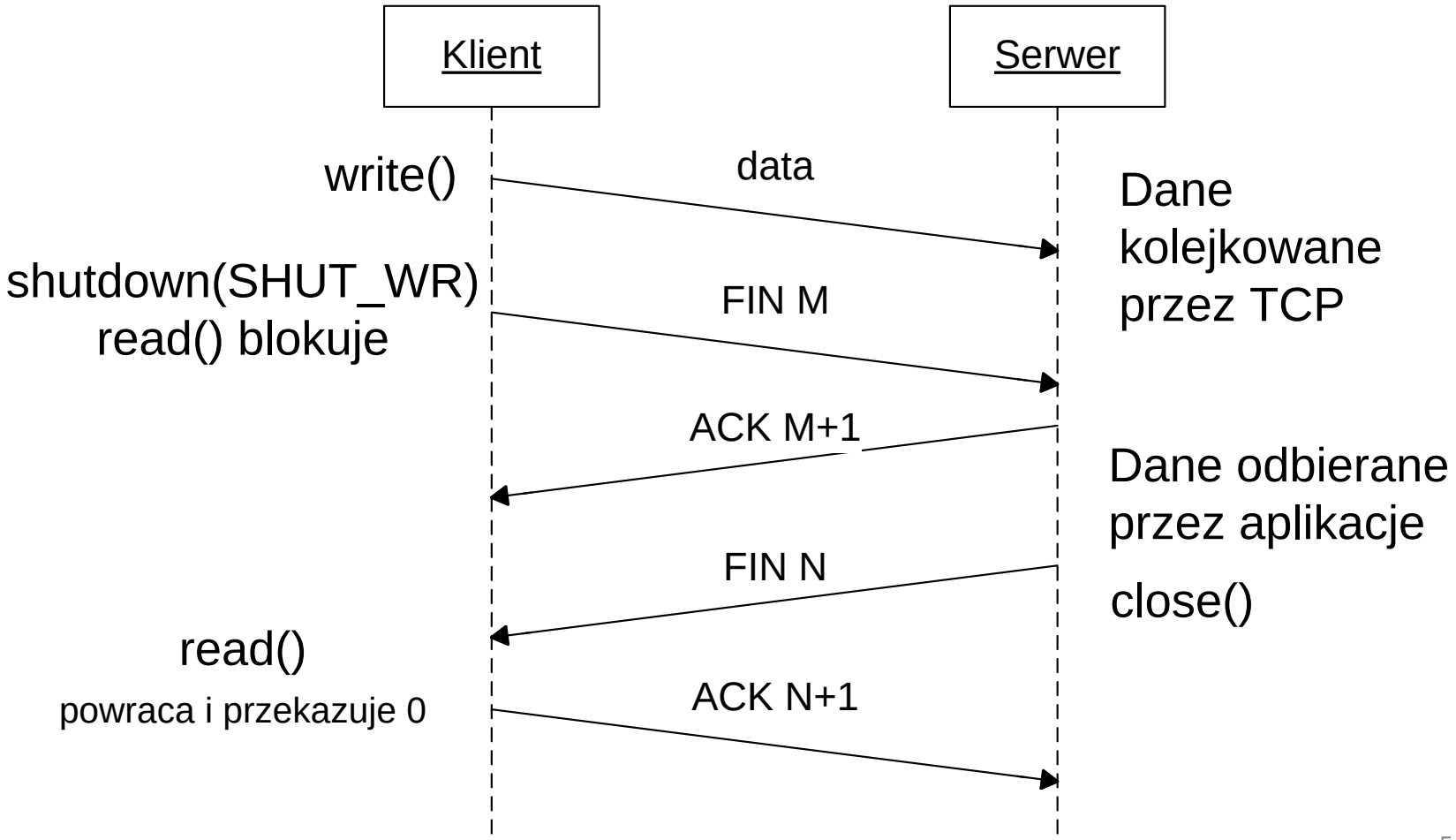


Opcja SO_LINGER ustawiona – zerwanie połączenia TCP przez wysłanie segmentu RST

```
l_onoff =1;
l_linger = 0;
```



Uzyskiwanie informacji o zakończeniu połączenia za pomocą funkcji shutdown()



close() summary

Function	Description
close(), l_onoff = 0	No more receives or sends can be issued on socket; contents of socket send buffer send to other end. If descriptor reference count becomes 0, then normal connection termination (FIN) sent following data in send buffer and socket receive buffer discarded;
close(), l_onoff = 1, l_linger = 0	No more receives or sends can be issued on socket; If descriptor reference count becomes 0, RST sent to other end; connection state set to CLOSED (no TIME_WAIT state; socket send buffer and socket receive buffer discarded.
close(), l_onoff = 1, l_linger != 0	No more receives or sends can be issued on socket; Contents of socket send buffer sent to other end. If descriptor reference count becomes 0, then normal connection termination (FIN) sent following data in send buffer; socket receive buffer discarded; and if linger time expires before connection CLOSED, close returns EWOULDBLOCK

shutdown() summary

Function	Description
shutdown(), SHUT_RD	No more receives can be issued on socket; process can still send on socket; socket receive buffer discarded; any further data received is discarded by TCP; no effect on socket send buffer.
shutdown(), SHUT_WR	No more sends can be issued on socket; process can still receive on socket; contents of socket send buffer send to other end, followed by normal connection termination (FIN); no effect on socket receive buffer;
shutdown(), SHUT_RDWR	No more receives or sends can be issued on socket; contents of socket send buffer send to other end. Normal connection termination (FIN) sent following data in send buffer and socket receive buffer discarded;

Opcje gniazd - podsumowanie

- Sposoby pobierania i ustanawiania opcji gniazd:
  - Funkcja getsockopt()
  - Funkcja setsockopt()
- Główne typy opcji:
  - Dla gniazd – SOL_SOCKET
  - Dla IPv4 – SOL_IP (IPPROTO_IP)
  - Dla IPv6 – SOL_IPV6 (IPPROTO_IPV6)
  - Dla TCP – SOL_TCP (IPPROTO_TCP)
- Parametry domyślne opcji gniazd można ustawić przez system plików **/proc** (parametry jądra)