# Programowanie sieciowe
## Stream Control Transmission Protocol

Michał Kalewski


Instytut Informatyki
Politechnika Poznańska
ul. Piotrowo 2, 60–965 Poznań

mkalewski@cs.put.poznan.pl
http://www.cs.put.poznan.pl/mkalewski

Poznań · 7 maja 2019 r.

## Plan wykładu





# Stream Control Transmission Protocol



## Stream Control Transmission Protocol (1/2)

*Stream Control Transmission Protocol* (SCTP) to uniwersalny protokół warstwy transportowej opisany w dokumencie RFC 4960.

Protokół SCTP zaproponowany został początkowo w roku 2000 w dokumencie RFC 2960.

Pakiety protokółu SCTP mogą być transmitowane protokołami IPv4 i IPv6.

Implementacje tego protokołu dostępne są m.in. w systemach operacyjnych UNIX, GNU/Linux i Cisco IOS; dla systemów operacyjnych Windows dostępne są implementacje zewnętrzne:

- SctpDrv (http://www.bluestop.org/SctpDrv/) ?,
- sctplib (http://www.sctp.de/sctp-download.html),
- SCTP for Node.js (https://www.npmjs.com/package/sctp/).

Implementacja dla systemów operacyjnych GNU/Linux rozwijana jest przez grupę *lksctp* (ang. *Linux Kernel Stream Control Transmission Protocol*) — http://lksctp.sourceforge.net (https://github.com/sctp/lksctp-tools).

## Stream Control Transmission Protocol (2/2)

Cechy protokołu SCTP:

- wykorzystuje komunikację strumieniową (dla pojedynczych *asocjacji*) lub pakietową/datagramową (dla wielu *asocjacji*);
- zawiera mechanizmy zwiększające bezpieczeństwo i niezawodność komunikacji;
- umożliwia realizację komunikacji zachowującej lub niezachowującej (ang. *ordered/unordered delivery*) kolejność FIFO pakietów;
- umożliwia realizację komunikacji łączem pojedynczym lub wieloma łączami (ang. *multi-homing*);
- umożliwia realizację komunikacji jedno- lub wielostrumieniowej (ang. *multi-streaming*);
- pozwala na utrzymywanie wielu *asocjacji* na pojedynczym gnieździe sieciowym.

Terminologia protokołu: *połączenie*, *asocjacja* oraz *strumień*.



## Zestawianie oraz zamykanie połączenia

Protokół TCP, do zestawienia połączenia, wykorzystuje tzw. procedurę *three-way handshake*, która wymaga wymiany trzech komunikatów: `SYN`, `SYN-ACK` oraz `ACK`.

Protokół SCTP, do zestawienia połączenia, wykorzystuje tzw. procedurę *four-way handshake*, która wymaga wymiany czterech komunikatów: `INIT`, `INIT-ACK`, `COOKIE-ECHO`, `COOKIE-ACK` (przeciwdziałanie atakom DDOS z użyciem pakietów `SYN`).

Protokół TCP, do zamykania połączenia, wykorzystuje procedurę wymiany dwóch/czterech komunikatów: `FIN` oraz `ACK`, co może prowadzić do połączeń „pół-otwartych”.

Protokół SCTP, do zamykania połączenia, wykorzystuje procedurę wymiany trzech komunikatów: `SHUTDOWN`, `SHUTDOWN-ACK`, `SHUTDOWN-COMPLETION` (przeciwdziałanie połączeniom „pół-otwartych”).



## Komunikacja wieloma łączami (*multi-homing*)

Protokół SCTP pozwala na zestawienie połączenia pomiędzy komunikującymi się procesami z użyciem wielu interfejsów sieciowych jednocześnie.

Możliwe jest także usuwanie i dodawanie interfejsów do istniejących już połączeń.

Niezawodność komunikacji poszczególnymi interfejsami sieciowymi weryfikowania jest wbudowanym mechanizmem *heartbeat*.

W przypadku awarii/wyłączenia wykorzystywanego aktualnie interfejsu sieciowego, protokół SCTP samoczynnie przełączy się na inny interfejs sieciowy zestawionego połączenia (transparentnie dla komunikujących się procesów).

Protokół SCTP zarządza automatycznie dostępnymi dla połączeń interfejsami sieciowymi, lecz możliwe jest wskazanie interfejsu preferowanego.



## Komunikacja wieloma strumieniami (*multi-streaming*)

W protokole SCTP, w ramach pojedynczego połączenia możliwe jest utworzenie wielu niezależnych strumieni komunikacyjnych (identyfikowanych numerami).

Liczba strumieni połączenia jest ustalana podczas zestawiania połączenia na podstawie deklaracji obu komunikujących się stron.

Porządkowanie pakietów w każdym ze strumieni jest niezależne.

Opóźnienia komunikacyjne (powodowane koniecznością retransmisji danych) w jednym strumieniu nie wpływają na komunikację w innych strumieniach.

## Gniazda sieciowe wielu asocjacji

Pojedyncze gniazdo sieciowe protokołu SCTP pozwala na zestawienie połączenia z wieloma zdalnymi procesami jednocześnie — każde z tych połączeń nazywane jest asocjacją.

Każda asocjacja identyfikowana jest numerem nazywanym *Association ID*.

Utrzymywanie wielu asocjacji na pojedynczym gnieździe sieciowym możliwe jest w procesach serwerów i klientów.

Pozwala to na realizację procesów komunikujących się w schematach 1-1, 1-N/N-1 oraz N-N bez konieczności wykorzystywania pod-procesów.



# Protokół SCTP w systemach operacyjnych GNU/Linux



## Protokół SCTP w systemach operacyjnych GNU/Linux (1/3)

Protokół SCTP wspierany jest przez jądra systemu operacyjnego GNU/Linux od wersji 2.4.

Tworzenie aplikacji własnych z użyciem tego protokołu może jednak wymagać instalacji dwóch dodatkowych pakietów: `libsctp-dev` oraz `lksctp-tools`.

Po zainstalowaniu powyższych pakietów dostępny jest program testowy `checksctp`, który weryfikuje czy jądro systemu obsługuje protokół SCTP.

Przykład:

```
$ checksctp
SCTP supported
```

Implementacja protokołu SCTP zawiera programy `sctp_darn` i `sctp_test` umożliwiające wysyłanie i odbieranie wiadomości w celach testowych.



## Protokół SCTP w systemach operacyjnych GNU/Linux (2/3)

Przykład (serwer):

```
$ sctp_darn -H 0 -P 2500 -l
sctp_darn listening...
Recieved SCTP_COMM_UP
SNDRCV
   sinfo_stream   0
   sinfo_ssn      0
   sinfo_flags    0x0
   sinfo_ppid     0
   sinfo_context  0
   sinfo_tsn      1576525018
   sinfo_cumtsn   0
   sinfo_assoc_id 2
DATA(6):  test.
Recieved SCTP_SHUTDOWN_COMP
```

Przykład (klient):

```
$ sctp_darn -H 0 -P 2600 \
> -h 127.0.0.1 -p 2500 -s
sctp_darn ready to send...
0:2600-127.0.0.1:2500> test
Recieved SCTP_COMM_UP
New connection, peer addresses
127.0.0.1:2500
192.168.1.1:2500
0:2600-127.0.0.1:2500>
```

*zob.* `sctp_darn(1)`, `sctp_test(1)`

## Protokół SCTP w systemach operacyjnych GNU/Linux (3/3)

Komunikacja TCP może być tunelowana protokołem SCTP.

W ramach narzędzi protokołu SCTP w systemach operacyjnych GNU/Linux dostępny jest program `withsctp(1)`, który pozwala na tego typu tunelowanie.

PRZYKŁAD (klient):

```
$ withsctp telnet HOST PORT
```



# Gniazda sieciowe protokołu SCTP



## Gniazda sieciowe protokołu SCTP

Gniazda sieciowe protokołu SCTP można utworzyć na dwa sposoby — różnica pomiędzy nimi dotyczy liczby jednoczesnych asocjacji, które będą przez nie wspierane.

Gniazda sieciowe protokołu SCTP wspierające tylko pojedyncze asocjacje (*gniazda strumieniowe*) tworzone są w sposób następujący:

```
int socket(PF_INET, SOCK_STREAM, IPPROTO_SCTP);
```

Gniazda sieciowe protokołu SCTP wspierające wiele jednoczesnych asocjacji (*gniazda datagramowe*) tworzone są w sposób następujący:

```
int socket(PF_INET, SOCK_SEQPACKET, IPPROTO_SCTP);
```

*zob.* `sctp(7)`



## Wymienność protokołów TCP i SCTP

Prosta zmiana protokołu transportowego TCP na protokół SCTP wymaga najczęściej jedynie zmiana argumentu wywołania funkcji systemowej `socket(2)`.

```
1 sfd = socket(PF_INET, SOCK_STREAM,
2 #ifdef USE_SCTP
3   IPPROTO_SCTP
4 #else
5   IPPROTO_TCP
6 #endif
7   );
```

## </> Komunikacja wieloma łączami (*multi-homing*) — selekcja łączy

```c
struct sockaddr_in addr, *addresses;
int addrlen = sizeof(struct sockaddr_in);
addresses = malloc(addrlen * 2);
addr.sin_family = AF_INET;
addr.sin_addr.s_addr = inet_addr(argv[1]);
addr.sin_port = 0;
bind(sfd, (struct sockaddr*) &addr, addrlen);
getsockname(sfd, (struct sockaddr*) &addr, &addrlen);
port = addr.sin_port;
memcpy(addresses, &addr, addrlen);
addr.sin_family = AF_INET;
addr.sin_addr.s_addr = inet_addr(argv[2]);
addr.sin_port = port;
memcpy(addresses + 1, &addr, addrlen);
sctp_bindx(sfd, (struct sockaddr*) addresses, 2,
           SCTP_BINDX_ADD_ADDR);
```



## Struktura `sctp_sndrcvinfo`

```c
struct sctp_sndrcvinfo {
  __u16 sinfo_stream;
  __u16 sinfo_ssn;
  __u16 sinfo_flags;
  __u32 sinfo_ppid;
  __u32 sinfo_context;
  __u32 sinfo_timetolive;
  __u32 sinfo_tsn;
  __u32 sinfo_cumtsn;
  sctp_assoc_t sinfo_assoc_id;
};
```

*zob.* `netinet/sctp.h`

https://github.com/torvalds/linux/blob/master/include/uapi/linux/sctp.h



## </> Komunikacja wieloma strumieniami (*multi-streaming*)

```c
int flags;
struct sctp_sndrcvinfo sndrcvinfo;
struct sctp_event_subscribe events;

memset((void*) &events, 0, sizeof(events));
events.sctp_data_io_event = 1;
setsockopt(sfd, SOL_SCTP, SCTP_EVENTS,
           (const void*) &events, sizeof(events));

sctp_sendmsg(sfd, (void*) buf, (size_t) sizeof(buf),
             NULL, 0, 0, 0, 1, 0, 0);

sctp_recvmsg(sfd, (void*) buf, sizeof(buf),
             (struct sockaddr*) NULL, 0,
             &sndrcvinfo, &flags);
```



## </> Gniazda sieciowe wielu asocjacji

```c
sfd = socket(...);
bind(sfd, ...);
listen(sfd, ...);
while(1) {
  rc = sctp_rcvmsg(sfd, ..., buf, ..., &info,  ...);
  assoc_id = info.info_assoc_id;
  stream = info.sinfo_stream;
  handle_message(assoc_id, stream, buf, rc);
}
```

# Podsumowanie



## Podsumowanie

- Protokół SCTP:
  - zestawianie i zamykanie połączenia;
  - komunikacja wieloma łączami (*multi-homing*);
  - komunikacja wieloma strumieniami (*multi-streaming*);
  - gniazda sieciowe wielu asocjacji.
- Narzędzia protkołu SCTP w systemach operacyjnych GNU/Linux.
- Gniazda sieciowe protokołu SCTP:
  - wymienność protokołów TCP i SCTP;
  - realizacja komunikacji wieloma łączami (*multi-homing*);
  - realizacja komunikacji wieloma strumieniami (*multi-streaming*);
  - realizacja gniazd sieciowych wielu asocjacji.



# Pytania i zadania



## Pytania i zadania — praca własna

1. Jaki jest maksymalny rozmiar danych, jaki można wysłać protokołem warstwy transportowej i od czego jest on uzależniony?
2. Czy te maksymalne rozmiary danych różnią się dla protokołów UDP, TCP oraz SCTP?
3. Czy w przypadku protokołu TCP można także mówić o asocjacji i ewentualnie co ona wówczas określa?

Literatura dodatkowa:

Sockets API Extensions for the Stream Control Transmission Protocol (RFC 6458):
https://tools.ietf.org/html/rfc6458

**Dziękuję za uwagę!**