Programowanie sieciowe
Buforowanie w komunikacji sieciowej

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 · 21 maja 2019 r.

Plan wykładu

Komunikacja strumieniowa i datagramowa

Komunikacja strumieniowa i datagramowa (1/2)

W komunikacji *datagramowej* (lub inaczej: *pakietowej*) niepodzielną jednostką wymiany danych jest datagram o rozmiarze określanym przez nadawcę lecz nie większym niż z góry znany rozmiar maksymalny.

Datagram może podlegać fragmentacji tylko przez warstwy niższe.

Z punktu widzenia programisty, w komunikacji datagramowej, pojedynczej operacji wysłania danych (np. jednemu wywołaniu funkcji systemowej `write(2)`) towarzyszy co najwyżej jedna operacja odebrania danych (np. jedno wywołanie funkcji systemowej `read(2)`).

Niezawodność transmisji i kolejność odbierania datagramów zależne są od konkretnego protokołu (*por.* UDP i SCTP).

Komunikacja strumieniowa i datagramowa (2/2)

W komunikacji *strumieniowej* niepodzielną jednostką wymiany danych jest pojedynczy bajt.

Z punktu widzenia programisty, w komunikacji strumieniowej, pojedynczej operacji wysłania danych (np. jednemu wywołaniu funkcji systemowej `write(2)`) może towarzyszyć konieczność wykonania wielu operacji odebrania danych (np. wiele wywołań funkcji systemowej `read(2)`).

Możliwa jest także sytuacja odwrotna, tj. wielu operacjom wysłania danych (np. wielu wywołaniom funkcji systemowej `write(2)`) może towarzyszyć pojedyncza operacja odebrania tych wszystkich danych (np. jedno wywołanie funkcji systemowej `read(2)`).

Internetowe protokoły komunikacji strumieniowej zapewniają niezawodność transmisji oraz kolejność FIFO odbierania danych (*por.* TCP i SCTP — możliwość komunikacji bez kolejności FIFO).

Bufory nadawcze i odbiorcze

Bufory nadawcze i odbiorcze

Sekwencja występowania buforów podczas komunikacji pomiędzy procesami nadawcy i odbiorcy:

1. *Przestrzeń użytkownika*: *bufor* w procesie nadawcy.
2. *Przestrzeń jądra systemu operacyjnego*: *bufor nadawczy* gniazda sieciowego.
3. TRANSMISJA DANYCH SIECIĄ KOMPUTEROWĄ.
4. *Przestrzeń jądra systemu operacyjnego*: *bufor odbiorczy* gniazda sieciowego.
5. *Przestrzeń użytkownika*: *bufor* w procesie odbiorcy.

Rozmiary tych wszystkich buforów mogą być różne!

Ograniczenia rozmiarów danych w komunikacji sieciowej

- Protokół TCP definiuje parametr MSS (ang. *maximum segment size*), który określa maksymalny rozmiar segmentu jaki nadawca może nadać; maksymalna wartość tego parametru to 65535 (2-bajtowe pole).
- Maksymalny rozmiar datagramu UDP wynosi 65535 bajtów włączając w to nagłówek (2-bajtowe pole długość).
- Maksymalny rozmiar pakietu IPv4 wynosi 65535 bajtów włączając w to nagłówek (2-bajtowe pole długość).
- Maksymalny rozmiar pakietu IPv6 wynosi 65575 bajtów włączając w to nagłówek (2-bajtowe pole długości danych).
- Ramki warstwy łącza danych określają MTU (ang. *maximum transmission unit*); dla sieci Ethernet wynosi on 1500 bajtów.
- Najmniejsza wartość MTU na trasie od nadawcy do odbiorcy nazywana jest *path MTU*.

Fragmentacja do rozmiaru MTU wykonywana jest przez protokoły IPv4 i IPv6.

Bufory nadawcze i odbiorcze gniazd sieciowych (1/3)

W systemach operacyjnych GNU/Linux rozmiary buforów nadawczych i odbiorczych gniazd sieciowych komunikacji strumieniowej oraz datagramowej zapisane są w katalogu `/proc/sys/net/ipv4`:

PRZYKŁAD:

```
$ cat /proc/sys/net/ipv4/tcp_wmem
4096  16384 4194304
$ cat /proc/sys/net/ipv4/tcp_rmem
4096  87380 6291456
$ cat /proc/sys/net/ipv4/udp_mem
23712 31618 47424
$ cat /proc/sys/net/ipv4/udp_wmem_min
4096
$ cat /proc/sys/net/ipv4/udp_rmem_min
4096
```

zob. `tcp(7)`, `udp(7)`

</> Bufory nadawcze i odbiorcze gniazd sieciowych (2/3)

```c
int getbuffsize(int sfd, int buffname) {
  int s;
  socklen_t slt = (socklen_t)sizeof(s);
  getsockopt(sfd, SOL_SOCKET, buffname, (void*)&s, &slt);
  return s;
}
void buffsizes(int sfd, int *srb, int *ssb) {
  *srb = getbuffsize(sfd, SO_RCVBUF);
  *ssb = getbuffsize(sfd, SO_SNDBUF);
}
int tcp, tcprb, tcpsb, udp, udprb, udpsb;
tcp = socket(PF_INET, SOCK_STREAM, 0);
buffsizes(tcp, &tcprb, &tcpsb);
udp = socket(PF_INET, SOCK_DGRAM, 0);
buffsizes(udp, &udprb, &udpsb);
```


Bufory nadawcze i odbiorcze gniazd sieciowych (3/3)

Zmiana rozmiarów buforów gniazda sieciowego możliwa jest z użyciem funkcji systemowej `setsockopt(2)`, musi to jednak nastąpić przed wywołaniem funkcji systemowej `connect(2)` (proces klienta) lub `listen(2)` (proces serwera).

```c
int srb = 12345;
setsockopt(sfd, SOL_SOCKET, SO_RCVBUF, &srb, sizeof(srb));
int ssb = 12345;
setsockopt(sfd, SOL_SOCKET, SO_SNDBUF, &ssb, sizeof(ssb));
```

(Wartość `srb`/`ssb` jest podwajana).

zob. `getsockopt(2)`, `setsockopt(2)`

Implementacja wymiany danych z użyciem komunikacji strumieniowej

</> Wysyłanie i odbieranie danych

Pętla wysyłania danych w komunikacji strumieniowej:

```
1  int _write(int sfd, char *buf, int len) {
2    while (len > 0) {
3      int i = write(sfd, buf, len);
4      len -= i;
5      buf += i;
6    }
7  }
```

Pętla odbierania danych w komunikacji strumieniowej:

```
1  int _read(int sfd, char *buf, int bufsize) {
2    do {
3      int i = read(sfd, buf, bufsize);
4      bufsize -= i;
5      buf += i;
6    } while (???);
7  }
```


Strategie wymiany danych w komunikacji strumieniowej

- Transmisja poprzedzająca z informacją o rozmiarze danych do wysłania.
- Znak końca danych.
- Stały rozmiar danych wysyłanych.

Przykład — protokół HTTP

PRZYKŁAD:

```
HTTP/1.1 200 OK
Date: Mon, 24 May 2016 06:28:53 GMT
Server: Apache/2.2.14
Last-Modified: Wed, 22 Jul 2015 19:15:56 GMT
Content-Length: 53
Content-Type: text/html
Connection: Closed

<html>
<body>
<h1>Hello, World!</h1>
</body>
</html>
```


Bezpieczne zamykanie połączenia TCP

</> Komunikacja klient — serwer

```
fd = socket(PF_INET, SOCK_STREAM, 0);
connect(fd, (struct sockaddr*) &saddr, sizeof(saddr));
write(fd, buf, 1000000);  // returns 1000000
close(fd);
```

```
fd = socket(PF_INET, SOCK_STREAM, 0);
bind(fd, (struct sockaddr*) &saddr, sizeof(saddr));
listen(fd, 5);
sl = sizeof(caddr);
cfd = accept(fd, (struct sockaddr*) &caddr, &sl);
// write(cfd, "Hello client!\n", 13);
do {
  rc = read(cfd, buf, bufsize);
  bufsize -= rc;
  buf += rc;
} while(rc > 0);
```


Zamykanie gniazda sieciowego

- Funkcja systemowa `close(2)` (TCP RST).
- Funkcja systemowa `shutdown(2)` (TCP FIN).
- Opcja gniazd sieciowych `SO_LINGER`.

</> Klient — wersja poprawiona

```
fd = socket(PF_INET, SOCK_STREAM, 0);
connect(fd, (struct sockaddr*) &saddr, sizeof(saddr));
write(fd, buf, 1000000);  // returns 1000000
shutdown(fd, SHUT_WR);
while(1) {
  rc = read(fd, tmpbuf, sizeof(tmpbuf));
  if (rc < 0) {
      perror("reading");
      exit(1);
  }
  if (rc == 0)
      break;
}
close(fd);
```


Podsumowanie

Podsumowanie

- Komunikacja strumieniowa i datagramowa.
- Ograniczeni rozmiarów danych w komunikacji sieciowej.
- Bufory nadawcze i odbiorcze podstawowych gniazd sieciowych.
- Strategie wymiany danych w komunikacji strumieniowej.
- Bezpieczne zamykanie połączenia TCP.

Pytania i zadania

Pytania i zadania — praca własna

1. Jakie muszą być wymagania dotyczące buforów nadawczych i odbiorczych gniazd sieciowych protokołu SCTP (w którym komunikacja jest pakietowa i niezawodna)?
2. Sprawdź jakie są rozmiary buforów nadawczych i odbiorczych podstawowych gniazd sieciowych w systemach operacyjnych Windows i Android.
3. Sprawdź ile wynosi wartość MTU dla komunikacji bezprzewodowej Wi-Fi i porównaj tę wartość z maksymalnym rozmiarem pola danych w ramkach tych sieci.

Dziękuję za uwagę!