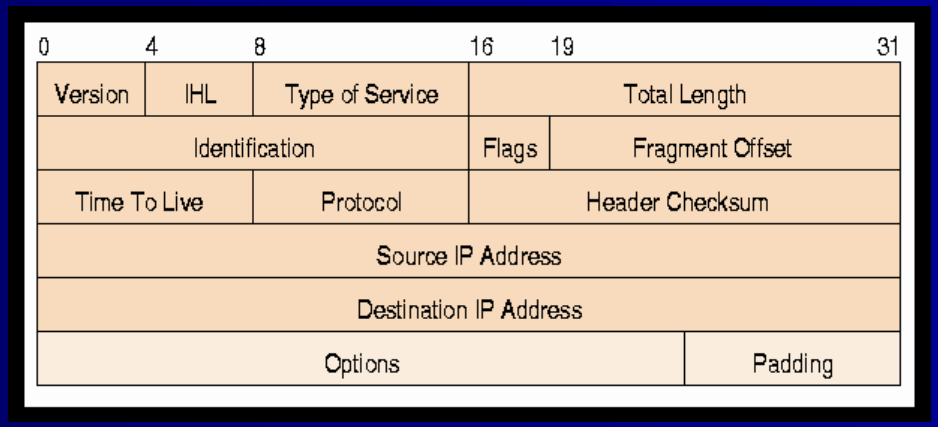
Ingineria Programării în Reţea (IPR)

Alegerea protocolului de transport

Internet Protocol (IP)

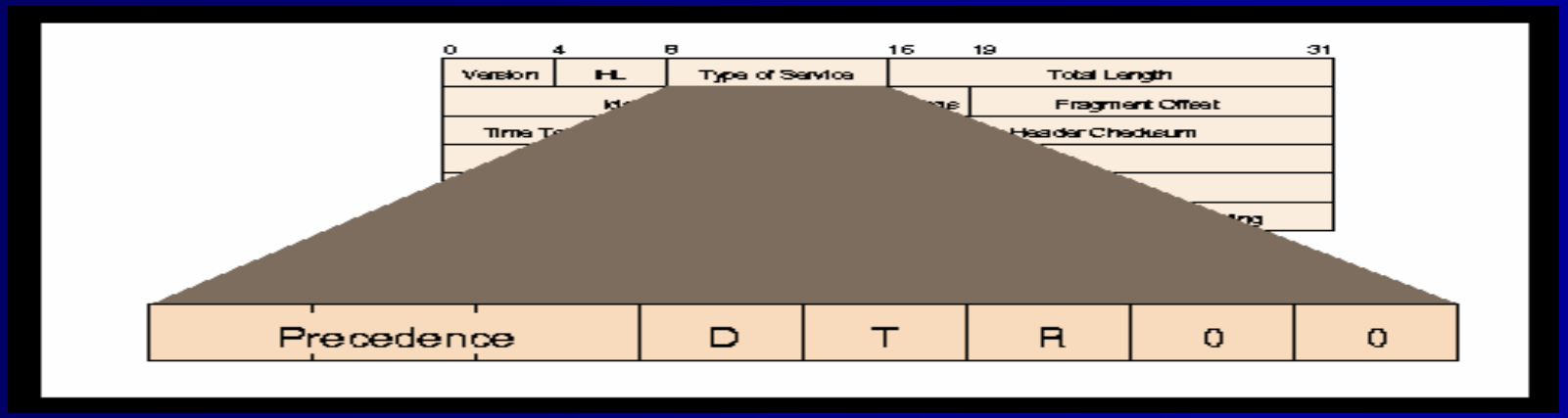
- Protocol de nivel 3, suport pentru toate protocoalele de transport de pe nivelul următor din stiva TCP/IP
- Descris in *Internet Protocol* (RFC 791)
- 2 versiuni: IPv4, IPv6

Header-ul IP



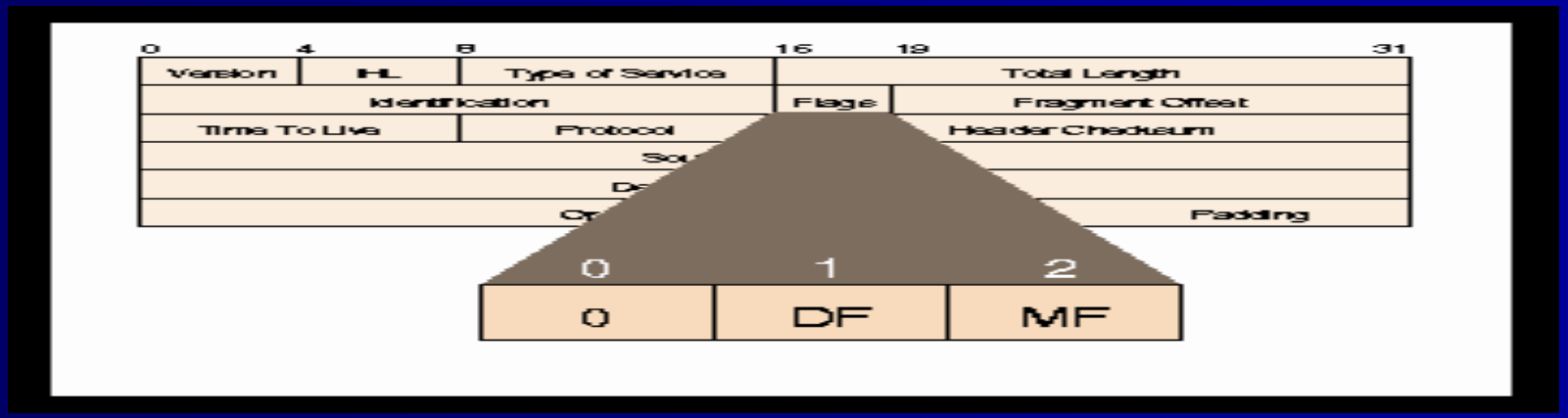
- ToS (Type of Service, RFC 1349) – "indicaţii" QoS privind serviciile cerute de aplicaţia generatoare a pachetului, pentru alte aplicaţii sau dispozitivele din reţea
- TTL (Time To Live) – pt. a preveni routarea "la infinit" a unui pachet şi irosirea resurselor

IP ToS "sugestii QoS"



Bits 0-2: Precedence.
Bit 3: 0 = Normal Delay, 1 = Low Delay.
Bit 4: 0 = Normal Throughput, 1 = High Throughput.
Bit 5: 0 = Normal Reliability, 1 = High Reliability.
Bit 6-7: Reserved for Future Use.

IP. Alţi biţi de “control”



Bit 0: reserved, must be zero

Bit 1: (DF) 0 = May Fragment, 1 = Don't Fragment.

Bit 2: (MF) 0 = Last Fragment, 1 = More Fragments.

UDP

- *User Datagram Protocol* (RFC 768)
- Oferă un serviciu de tip *"best-effort"* bazat pe datagrame
- Simplu, oferă libertate totală utilizatorului / aplicaţiei
- Nu implementează nici un mecanism de prevenţie a pierderii sau reordonării, şi nici nu îşi adaptează rata automat la condiţiile din reţea

Header-ul UDP

Table 1-33. User Datagram Protocol headers

0										1										2										3	
0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1
Source Port										Destination Port																					
Length										Checksum																					
Data																															

- Header-ul UDP nu conţine nici un câmp care să codifice vreo cerinţă sau recomandare d.p.d.v. QoS
- Conţine o sumă de control (checksum) pentru verificarea integrităţii datagramei

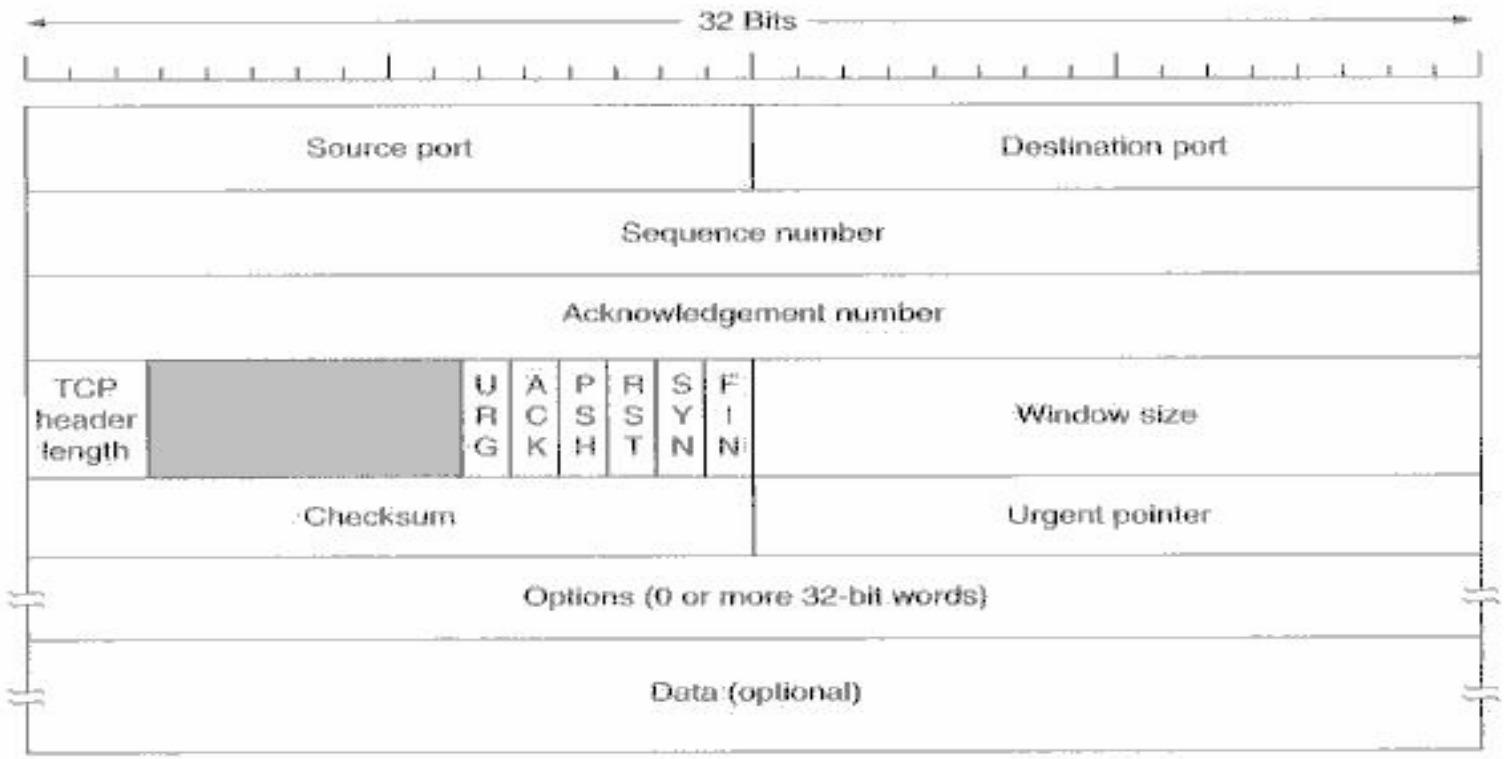
UDP QoS?

- Controlul aparţine exclusiv aplicaţiei / utilizatorului: orice mecanism de QoS (retransmitere, ordonare etc.) se va implementa la nivelul aplicaţiei

TCP

- *Transport Control Protocol* (RFC 793)
- Implementează mecanisme de confirmare a primirii pachetelor, de retransmisie, de evitare a congestiei
- Oficial evită congestia, ne-oficial... o crează, ocupând în timp toată banda disponibilă

Header-ul TCP

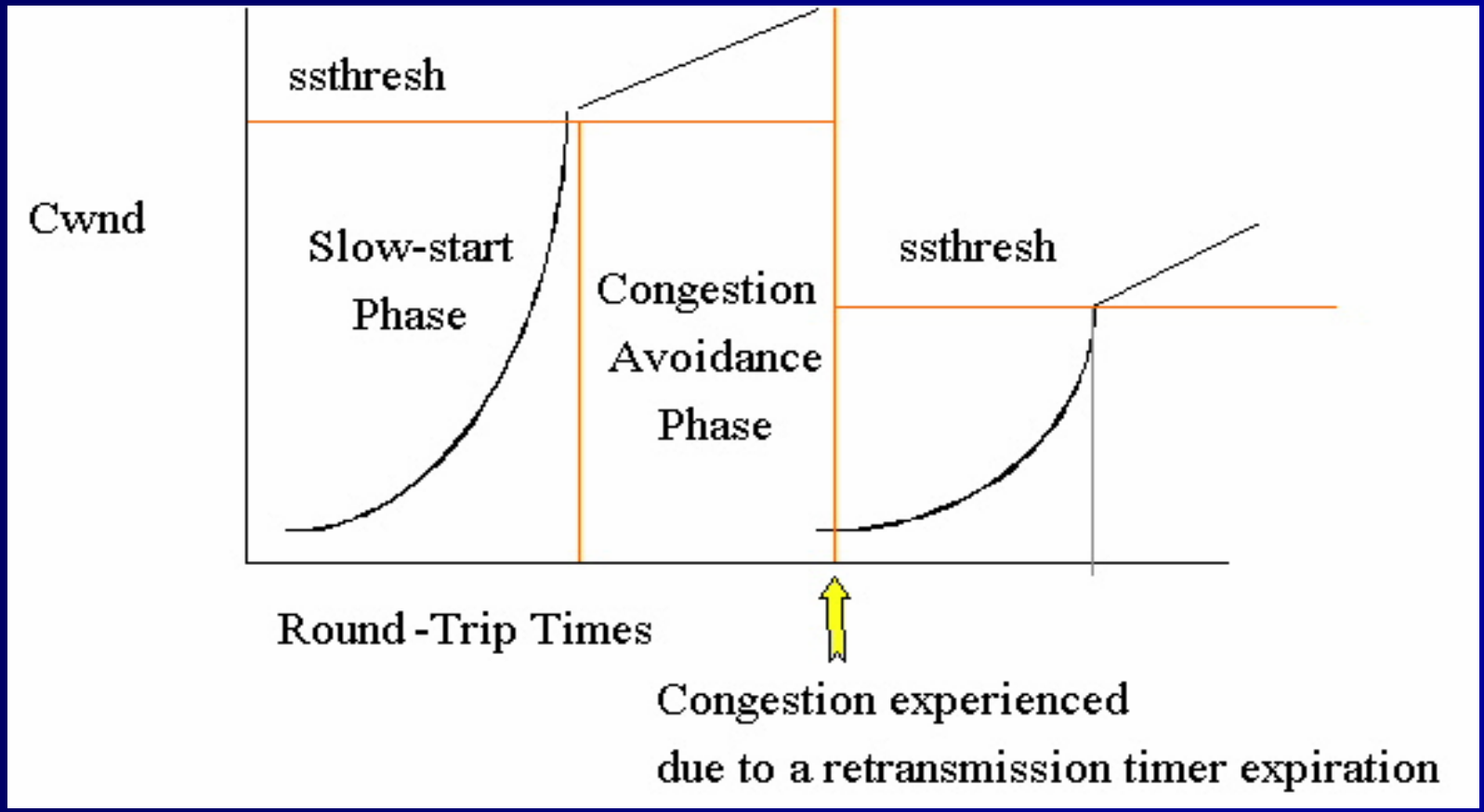


The TCP header.

TCP - faze

- "Slow" start
  - Fereastra de transmisiune a TCP creşte exponenţial de la 0 la ssthresh (Slow Start Threshold)
- Congestion avoidance
  - Fereastra de transmisiune creşte liniar, cu o rată de un segment / RTT

TCP – trafic "elastic"



Comparaţie TCP vs. UDP

TCP	UDP
Trafic de tip *ELASTIC* (mecanism de adaptare a ratei de transmisiune la condiţiile din reţea)	Trafic de tip *INELASTIC* (rata de transmisiune nu se adaptează la condiţiile din reţea
Garantează integritatea datelor (mecanism de confirmare a primirii)	Nu garantează integritatea datelor
Aplicaţii de tip transfer de fişiere, tranzacţii	Aplicaţii de tip streaming (audio, video)

TCP tuning

- Performanţele TCP-ului depind de
  - Performanţele HW (CPU, memorie, PCI)
  - Valorile parametrilor de configurare (TCP, interfaţă reţea, placă de bază etc.)

TCP tuning - parametri

- Parametrii interfeţei de reţea
  - Interrupt coalescence (configurabil la nivelul driverului interfeţei de reţea)
  - MTU (Maximum Transmission Unit)

```
#/sbin/ifconfig eth0 mtu 1500
```

  - Tx buffer length

```
#/sbin/ifconfig eth0 txqueuelen 2000
```

TCP tuning (2)

- Parametrii kernel Linux
  - Coada de recepţie TCP: `/sbin/sysctl -w sys.net.core.netdev_max_backlog=2000 (implicit 300)`
  - Pentru a preveni memorarea RTT-ului pentru o anumită conexiune: `/sbin/sysctl -w sys.net.ipv4.route.flush=1`
  - Confirmări selective: `/sbin/sysctl -w net.ipv4.tcp_sack=0`

TCP tuning (3)

- Dimensionarea bufferelor socket (kernel Linux): socket buffer size = 2 * bandwidth * delay (BDP – Bandwidth Delay Product)

```
int socket_descriptor, int sndsize;
err = setsockopt (socket_descriptor, SOL_SOCKET,
   SO_SNDBUF, (char*)&sndsize,
   (int)sizeof(sndsize));
int socket_descriptor, int rcvsize;
err = setsockopt (socket_descriptor, SOL_SOCKET,
   SO_RCVBUF, (char*)&rcvsize,
   (int)sizeof(rcvsize));
int sockbufsize=0; int size=sizeof(int);
err=getsockopt(socket_descriptor, SOL_SOCKET,
   SO_SNDBUF, (char*) &socketbufsize, &size);
```

TCP tuning (4)

- **Socket buffer memory queue limits (R/W)**

```
/sbin/sysctl -w net.core.rmem_max= VALUE
/sbin/sysctl -w net.core.wmem_max= VALUE
/sbin/sysctl -w net.ipv4.tcp_mem= MIN DEFAULT
MAX
```

Exemplu `socket` Python

```
import socket
print "Creating socket..."
s = socket.socket( socket.AF_INET,
  socket.SOCK_STREAM )
print "done."
print "Looking up port number..."
port = socket.getservbyname( 'http', 'tcp' )
print "done."
print "Connecting to remote host on port %d..." %
  port s.connect( ("www.google.com", port) )
print "done."
print "Connected from", s.getsockname()
print "Connected to", s.getpeername()
```