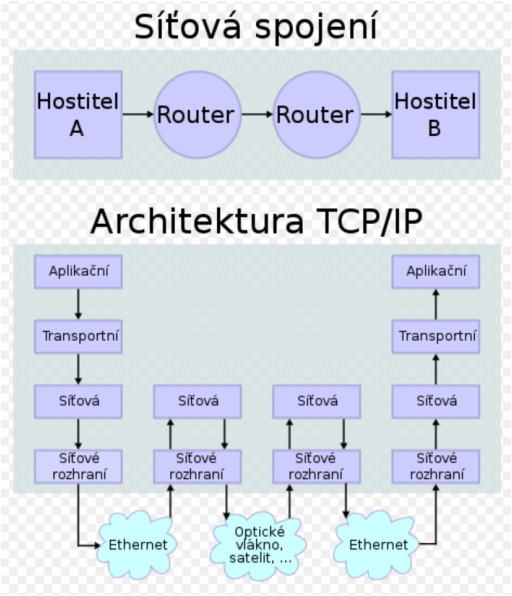
Transportní vrstva

Transportní vrstva (L4) **zajišťuje službu „end-to-end connection**“. Komunikace mezi cílovými aplikacemi a zajištění překlenutí požadavků vyšších vrstev vzhledem k možnostem vrstev vyšších (někdy se nazývá přizpůsobovací vrstva). Transportní vrstva je realizována v rámci realizace komunikace koncových uzlů (přenos dat mezi uzly je realizován pouze do L3 – síťová vrstva). **Všechny služby transportní vrstvy zajišťuje síťová část OS (nebo síťový SW).**



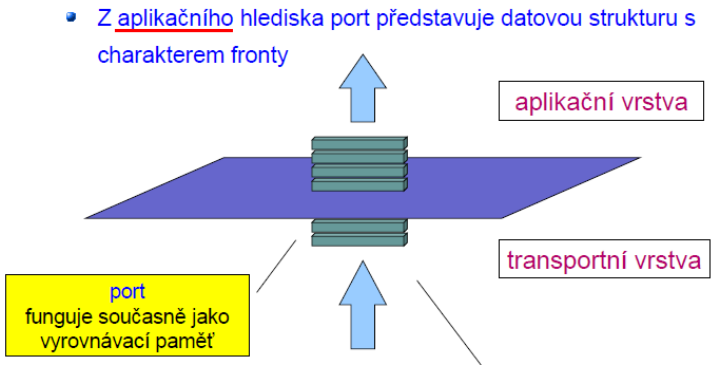
K zajištění služby plní transportní vrstva dvě funkce:

- **Multiplexing/demultiplexing dat** - pomocí většinou jedné komunikační cesty je potřeba data odebírat/předávat více aplikacím.
- **Poskytnutí požadované služby ohledně zabezpečení přenosu dat** – různé aplikace mají různé požadavky na rychlost a spolehlivost přenosu a doručení dat.

Multiplexing/demultiplexing dat

Komunikační cesta KZ je většinou jediná (síťové rozhraní), ale **aplikací současně běžících a komunikujících je veliké množství.** Je **potřeba přiřadit identifikaci dat dle aplikace**. Nepoužívá se identifikace konkrétně dle jednotlivých procesů, ale **dle „přechodových „ bodů – portů.**

Identifikace porty je softwarová logická záležitost. **Označení je univerzální** - na všech OS (platformách) se používají stejně. Port je určen 16bit číslem (0h- FFFFh, 0-65535). **Realizace portu záleží na platformě** (např. socketové API – viz.dále).



Pozn.: **Z hlediska programátora se jedná o buffer tipu FIFO**. Programátor se nemusí o naplňování bufferu moc starat.

Porty jsou dynamicky obsazovány. Procesy aplikací se takto připojují a odpojují od portů.

V přidělování čísel portů je potřeba **zachovat pořádek. Tuto funkci plní IANA** jakožto globální autorita.

Čísla portů jsou zhruba rozděleny na 3 skupiny.

- dobře známé porty 0-1023
- registrované porty 1024-49151
- ostatní 49152 – 65535

Přidělení portů aplikacím je dáno RFC a je zveřejněno v rámci „Protocol Assignments“ na https://www.iana.org/assignments/service-names-port-numbers/service-names-port-numbers.xhtml. Známé porty se nemění a na těchto portech „ naslouchají“ serverové části aplikací zda není klienty požadováno zahájení komunikace.

Dále je port určen typem transportního protokolu – TCP nebo UDP a tím je definována kompletně služba transportní vrstvy. Příklad přiřazení známých portů je níže:

20	tcp	udp	FTP (data)
21	tcp	udp	FTP (příkazy)
22	tcp	udp	SSH
23	tcp	udp	Telnet
25	tcp	udp	SMTP
37	tcp	udp	TIME protocol
49	tcp	udp	Login Host Protocol TACACS
53	tcp	udp	DNS
65	tcp	udp	TACACS - Database Service
66	tcp	udp	Oracle SQL*NET
67	tcp	udp	BOOTP (server), DHCP
68	tcp	udp	BOOTP (klient), DHCP
69	tcp	udp	TFTP
70	tcp		Gopher
79	tcp	udp	Finger
80	tcp	udp	HTTP
88	tcp	udp	Kerberos
109	tcp	udp	Post Office Protocol - Version 2
110	tcp	udp	Post Office Protocol - Version 3, POP3

Socketové API – realizace „aplikačního“ spojení

Aplikace (entity aplikační vrstvy) pracují s porty prostřednictvím aplikačního rozhraní (API – application programming interface). **API je součástí OS nebo je definováno pomocí knihoven aplikace.**

Ovládání portů založené na socketech pochází z BSD Unixu. Socket byl původně určen pro potřeby meziprocesové komunikace, který slouží pro předávání dat mezi různými procesy v rámci jednoho počítače. Umožňuje obousměrnou komunikaci mezi vstupy a výstupy. Socketové API emuluje procesům aplikací práci se sockety (např. WINSOCK) tak, že **je jedno jestli se jedná o „lokální“ komunikaci v rámci operačního systému nebo „síťovou“ komunikaci.**

Descriptory zakládají sockety a vznikají voláním procesu

„**descriptor=SOCKET(protocol family, typ služby, protocol)“,**

např.„descriptor = SOCKET (PF_INET, stream, TCP)“

(descriptor souboru/procesu – „popisovač“ - viz. např.OS LNX)

Sockety jsou přiřazeny k portům

„BIND=(socket, IDspojení)“

ID spojení je identifikace počátečního a koncového bodu spojení (body jednoho spojení jsou dva!).
Bod spojení je dán

<IPadd: číslo portu>

např. 85.25.45.52:1580

Otevřené komunikace se zobrazí pomocí univerzálního příkazu „**netstat“ (níže výpis „netstat –n“).**

```
C:\Users\Uživatel>netstat -n

Aktivní připojení

  Proto  Místní adresa          Cizí adresa            Stav
  TCP    10.0.0.139:50248       130.117.190.210:443    TIME_WAIT
  TCP    10.0.0.139:50249       130.117.190.195:443    TIME_WAIT
  TCP    10.0.0.139:50250       130.117.190.210:443    TIME_WAIT
  TCP    10.0.0.139:50251       130.117.190.210:443    TIME_WAIT
  TCP    10.0.0.139:50252       62.128.100.37:443      TIME_WAIT
  TCP    10.0.0.139:50253       62.128.100.37:443      TIME_WAIT
  TCP    10.0.0.139:50254       195.122.169.15:80      TIME_WAIT
  TCP    10.0.0.139:50255       130.117.190.210:443    TIME_WAIT
  TCP    10.0.0.139:50256       130.117.190.210:443    TIME_WAIT
  TCP    10.0.0.139:50257       130.117.190.195:443    TIME_WAIT
  TCP    10.0.0.139:50258       130.117.190.210:443    TIME_WAIT
  TCP    10.0.0.139:50259       130.117.190.195:443    TIME_WAIT
  TCP    10.0.0.139:50260       130.117.190.210:443    TIME_WAIT
  TCP    10.0.0.139:50261       130.117.190.210:443    TIME_WAIT
```

Pozn.: Socketové rozhraní je vhodné pro výpočetní model (architekturu) klient-server.

Další operace se sockety:

LISTEN (socket, queue..) - server naslouchá na připojeném portu
ACCEPT (socket,..) - volání, čeká na příchod požadavku na zadaném socketu (vytváří nový socket a vyplní descriptor požadavku – to je druhý směr komunikace)
CONNECT(socket, ID spojení..) – požadavek na straně klienta na spojení se serverem
SEND(socket, data, délka, flags)- posílání dat do navázaného spojení (při spojové komunikaci)
SENDTO(socket…..) -posílání dat při nespojové komunikaci
RECV(socket, buffer, délka, flags) – příjem dat ze socketu při spojové komunikaci
RECVFROM(…) - příjem dat ze socketu při nespojové komunikaci
CLOSE (socket) - ukončení spojení, zavření socketu, zrušení descriptoru

Typ transportní služby - poskytnutí požadované služby ohledně zabezpečení přenosu dat

Poskytované služby lze dle jejich typu rozdělit:

- **Stream** – spojová a spolehlivá služba (protokol TCP)
- **Datagram** – nespolehlivá a nespojová (datagramová)služba (protokol UDP)
- **RAW** – speciální režim, pseudoprotokoly transportní vrstvy (např. ICMP, IGMP)

Protokoly využívají nespolehlivý a nespojovaný přenos dat (IP), zajišťovaný síťovou vrstvou.

Pojmy:
Nespojová(datagramová)služba – je služba, kdy každý paket je samostatnou datovou jednotkou bez jakékoliv vazby na ostatní. Je směrován sítí nezávisle různými cestami. Služba je bezestavová. Přenos je co nejjednodušší, rychlý a efektivní.

Nespolehlivá služba – příjem dat není potvrzován. Není zajištěno seřazení datagramů a ani jejich doručení. Není zabezpečení doručení dat.

Spojová služba – během přenosu dat se vytvoří spojení (relace) mezi účastníky přenosu. Spojení vždy prochází stavy přenosu jako zahájení, přenos a ukončení. Posloupnost přechodů mezi stavy je definována. Nelze začít přenosem a potom pokračovat zahájením apod. Spojová služba je současně i stavovou službou.

Spolehlivá služba – příjem dat je zabezpečen (zajištěn). Příjem dat je potvrzován. Data jsou současně i seřazena (pakety se mohou, díky vlastnostem síťové vrstvy, různě předbíhat).

Srovnání protokolů UDP a TCP

UDP	TCP
Nespojovaná služba, mezi hostiteli není třeba vytvořit relaci.	Služba orientovaná na připojení, mezi hostiteli se vytváří relace.
Protokol UDP nezaručuje doručení dat, nezajišťuje potvrzování ani správné pořadí dat.	Protokol zaručuje dodání paketů s využitím potvrzování a sekvenčního přenosu dat.
Programy, které používají protokol UDP, musí samy zajistit spolehlivost potřebnou k přenosu dat.	Programům, které používají protokol TCP, je zaručena spolehlivost přenosu dat.
Protokol UDP je rychlý, s malými požadavky na objem provozních dat, a podporuje dvoustrannou komunikaci i komunikaci jednoho hostitele s více hostiteli.	Protokol TCP je pomalejší, má vyšší nároky na objem provozních dat a podporuje pouze dvoustrannou komunikaci.

Protokol UDP – User Datagram Protokol

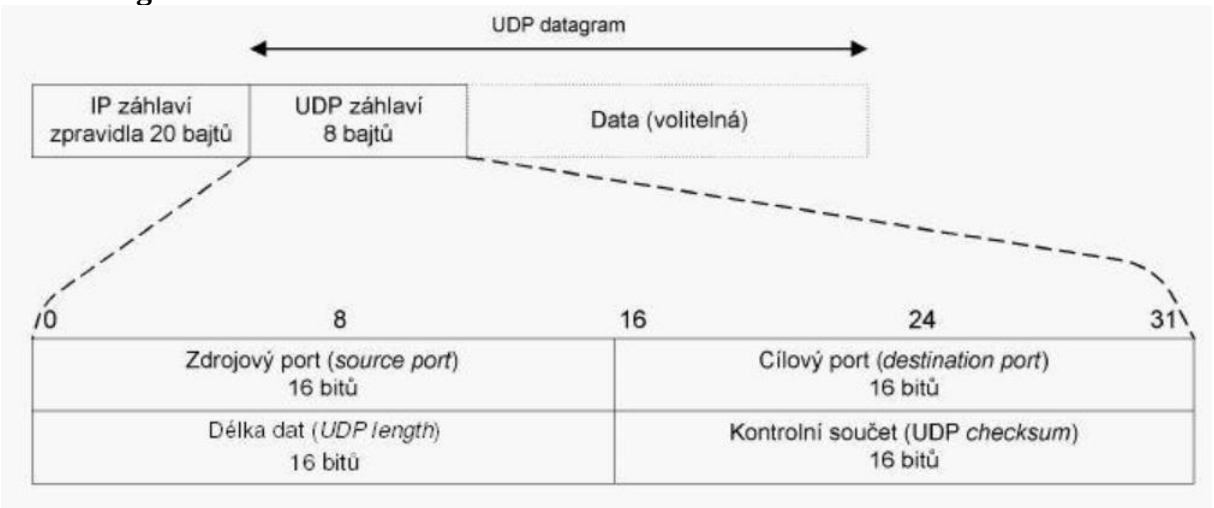
Je **datagramové řešení transportních služeb** – **nespolehlivé a nespojové**. Jedná se o co nejjednodušší nadstavbu nad síťovou vrstvou (ta je datagramová). Volbu úrovně **zabezpečení doručení dat nechává na aplikaci.**

Je **jednoduchý a rychlý** s minimální režií. Používá se v rámci **lokální komunikace (LAN),** kde k „přeházení“ nebo ztracení datagramů běžně nemůže dojít.

Je **ideální pro aplikace typu dotaz-odpověď(DNS dotaz**) nebo tam, kde jsou pouze přenášeny **krátké bloky dat do 512B (BootP, SNMP, RIP update, hry…)** nebo pro **real-time přenosy (multimediální přenosy, VoIP hovor ...),** kde malá ztráta dat není zásadní, na rozdíl od rychlosti přenosu dat.

Používá se pro zajištění broadcast nebo multicast vysílání (je to lokální záležitost), protože spojovým (stavovým) a spolehlivým (s potvrzováním přijetí dat) spojením TCP to nejde (RIP update). Množství aplikací (VoIP, DNS…) řeší služby dle jejich povahy, rozdělením na část zajišťovanou UDP, a část zajištěnou pomocí TCP.

UDP datagram



Hlavička má 8B a obsahuje:

- **porty** (source a destination port)

- **délka** v B
- **kontrolní součet** – jedná se o checksum UDP datagramu doplněného o „pseudohlavičku“ (část IP hlavičky – IP adresy, ID neseného protokolu a délka paketu). Je to pro ochranu proti nesprávně doručeným datagramům. Když je špatný kontrolní součet je UDP datagram zahozen (negeneruje se ICMP zpráva).Představuje to minimální stupeň zabezpečení správného přenosu dat.

Protokol TCP – Transmission Control Protocol

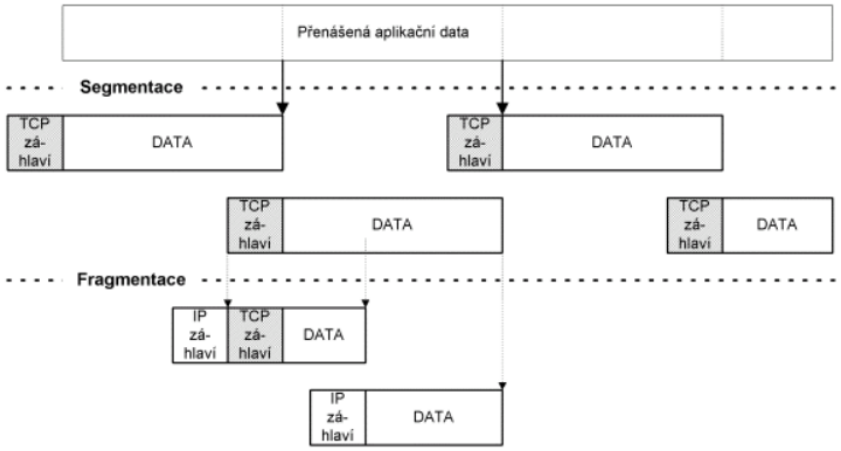
K zajištění **spojové služby** je potřeba

- **Stavový přenos**, který je vždy dvoubodový (jeden příjemce a jeden odesílatel). Tímto vzniká relace nebo vlákno spojení.
- **Plně duplexní přenos dat** - FD.
- **Streaming dat** – spojová služba vytváří zdání pevného spoje (leasen line) mezi KZ. K tomu je potřeba zajistit proudový přenos „byte po bytu“ – streaming dat (stream – viz. typ služby L4). Proud bytů je zajištěn pomocí segmentace dat (viz.dále). Aplikace spolu komunikují v iluzi „bytové roury“ – data jsou předávána po B.
- **Řízení toku dat** - protože spojení je provedeno nad datagramovým (nespolehlivým) prostředím a ani odezvy koncových zařízení nelze garantovat (např. www server). Rychlost toku dat je potřeba přizpůsobit aktuální situaci, aby nedošlo například k zahlcení.

K zajištění spolehlivé služby je potřeba

- **Potvrzení doručení dat** – informace o správném doručení a seřazení dat. To se provádí pomocí kladného a okénkového potvrzování.

Streaming dat - segmentace dat – TCP segmenty



Datová jednotka přenášená protokolem TCP se nazývá **TCP segment**. Data vysílaná aplikací jsou převáděna pomocí bufferu do jednotlivých datových segmentu. Vyrovnávací paměť (buffer) je realizována části paměti identifikované portem. Kratší bloky dat (např. terminálový provoz) je možné **vynuceně vyslat z bufferu pomocí mechanismu „TCP push“**(řídící bit PUSH). Znaky se doplní výplní a odešlou jako segment dat. Po doplnění TCP hlavičky se nazývá blok dat TCP segment.

Aplikace posílá data do vyrovnávací paměťi neustále s proměnnou rychlostí (rychlost vysílání lze regulovat pomocí řízení toku dat -velikosti časového okénka – viz. dále). **Vzniká iluze propojení aplikací pevnou linkou s určitými parametry**. Takto je realizován **streaming dat po bytech**.

Potvrzení doručení, seřazení dat a řízení toku

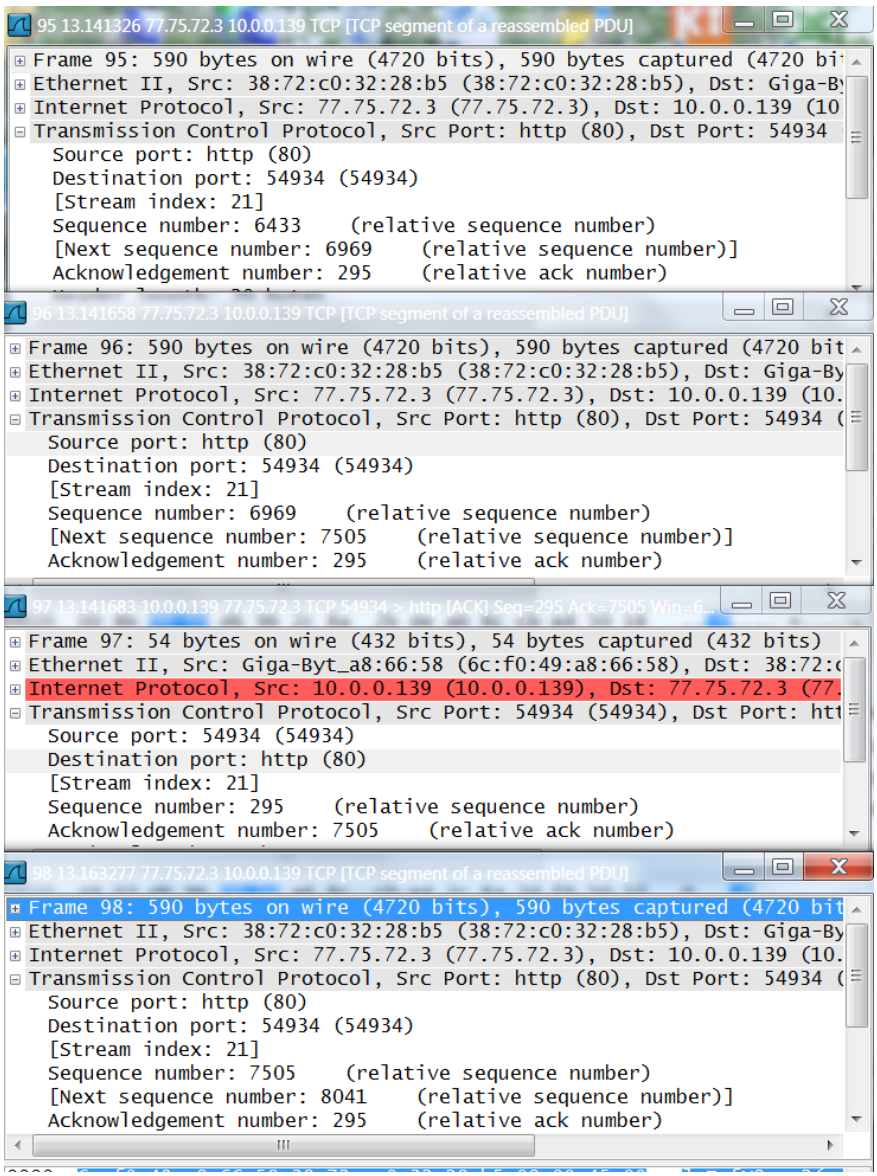
TCP používá **kladné okénkové potvrzování dat**.

Při kladném potvrzování se potvrzují pouze správně přijatá data. Kladné potvrzování v TCP **realizováno parametry sequence (sekvenční)number dále SN a acknowledgement number dále ACKN.**

Jedná se o **32 bitová čísla** určující vyslané a přijaté byty. **Počáteční hodnota je „plovoucí“** (není pevně dána, většinou „0“). Podstatné je potom „vázání“ údajů v řadě vyslaných dat. **SN určuje „pořadové“ číslo prvního bytu vysílaného segmentu dat. ACKN je potvrzením „pořadového“ čísla posledního přijatého bytu segmentu dat. Čísla na sebe navazují.**

Přenos dat je full duplex a proto v jednom TCP segmentu je SN pro data vysílaná (jeden směr dat) a ACKN pro data přijímaná (druhý směr dat). V reálném provozu modelu klient-server je po ustanovení spojení potvrzení v rámci přenosu dat doprovázeno řídícím bitem ACK. **Další SN vysílaného segmentu dat navazuje na přijaté ACKN .**

V rámci časového intervalu nemusí být potvrzován každý segment dat. Vychází se z posledního přijatého „v spojité řadě bytů “ na který navazuje další SN. Jestliže není některý segment doručen, vrací se poslední přijatý byte z řady a od tohoto místa se vysílání opakuje. Jestliže dojde k „předběhnutí“ segmentů, opět se takto vrací poslední „spojitě“ přijatý byte a data se seřadí dle SN nebo se opakují tak, aby došlo k jejich seřazení. Obrázek ukazuje část komunikace - dva segmenty, jejich potvrzení a následující segment dat.



Dle okamžiku potvrzení lze **rozdělit potvrzování vázané či nevázané** například tzv. **okénkové**.

Vázané potvrzování je spjato s **čekáním na potvrzení o přijetí dat před odesláním dalšího segmentu dat.** Výhodné je to, že není potřeba data nikde ukládat do vyrovnávací paměti a potvrzený segment již není potřeba. Podstatnou **nevýhodou je různá doba čekání na potvrzení doručení.** Tato doba je neefektivní a **přenos dat by se neúměrně prodloužil. Proto se tento typ nepoužívá**.

TCP používá nevázané potvrzování pomocí časového okénka. Data jsou potvrzována se zpožděním v časovém intervalu daným velikostí časového okénka. Odeslané nepotvrzené segmenty je potřeba proto uložit ve vyrovnávací paměti do doby potvrzení (mírná nevýhoda).

Velikost časové okénka se mění dle okamžitého stavu celé přenosové cesty. Pro nastavení okénka se měří při každé „obrátce“ data – potvrzení **parametr RTT (Round Trip Time** – typicky ms - µs). Z času obrátky se vypočítá „timeout“ jako vážený průměr RTT a z toho se odvozuje velikost časového okénka. **Tímto TCP docílí řízení rychlosti toku dat.** Časové **okénko se neustále posouvá podle odeslaných segmentů s časem**, a proto se mu také říká **časové posuvné okénko.** Přenos je full duplexní a proto každý směr má svoje parametry (pro aplikace klient-server se ale jedná většinou o jeden směr v jednom čase).

Okénko je realizováno v TCP hlavičce parametrem „Window size“ a má 16bitů. Hodnota „0“ představuje „zavření“ okna a zastavení přenosu dat. Spojení zůstává zachováno a přenos může kdykoliv začít. **Maximální hodnota 65535 pak představuje „plně otevřené“ okno a přenos může být co nejrychlejší**. Tímto se prodlužuje časové okno na maximum a přenáší se největší množství segmentů dat bez potvrzení.

Spojová služba

V rámci TCP se navazuje „spojení“ mezi aplikacemi. Přenos dat je realizován v rámci tohoto spojení. Vláken TCP spojení je většinou navázáno více (např. aplikace ftp – řídící a datové).

Spojení je v každém okamžiku definováno jeho stavem. Proto také stavová služba. Stavy spojení jsou:

- Zahájení
- Přenos dat
- Ukončení

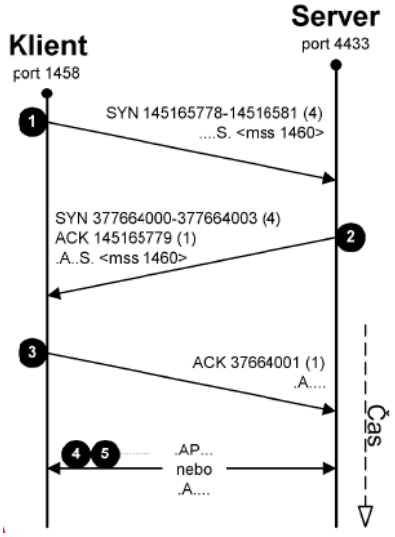
TCP protokol využívá pro řízení stavu spojení řídící bity (control bits). Jsou to

- SYN
- ACK
- RST
- FIN

Další bity jsou použity pro již popsané funkce

- URG
- PSH

Zahájení spojení
Vždy se používá „ Three-Way handshake“. Tato posloupnost je pouze při zahájení. Níže je uvedený příklad nejběžnější komunikace mezi klientem a serverem.



1. **Klient začíná navazovat spojení**

TCP paket 1, připojuje se na port 4433

nastaven SYN

SN = náhodné číslo v rozsahu (0, 2^{32}-1)

1 je prvním segmentem - nemůže potvrzovat žádná data - není nastaven ACK

2. **Server potvrzuje spojení a navazuje zpět 2 spojení**

nastaven ACK

nastaven SYN

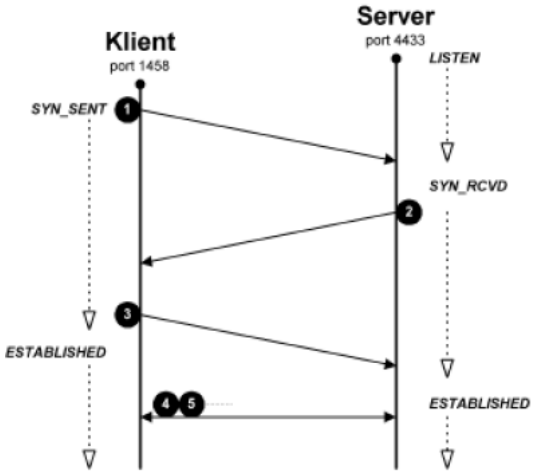
SN = náhodné číslo v rozsahu (0, 2^{32}-1)

3. **Klient potvrzuje 2. spojení**

nastaven ACK

Pozn.:mss - Maximum segment size

Příklad odpovídajících operací (stavů socketového API) při spojení



SYN_SENT
v době navazování spojení
LISTEN
server „naslouchá“ - očekává spojení
SYN_SEND
klient odeslal první TCP segment
SYN_RCVD
server dostal první TCP segment
ESTABLISHED
spojení navázáno

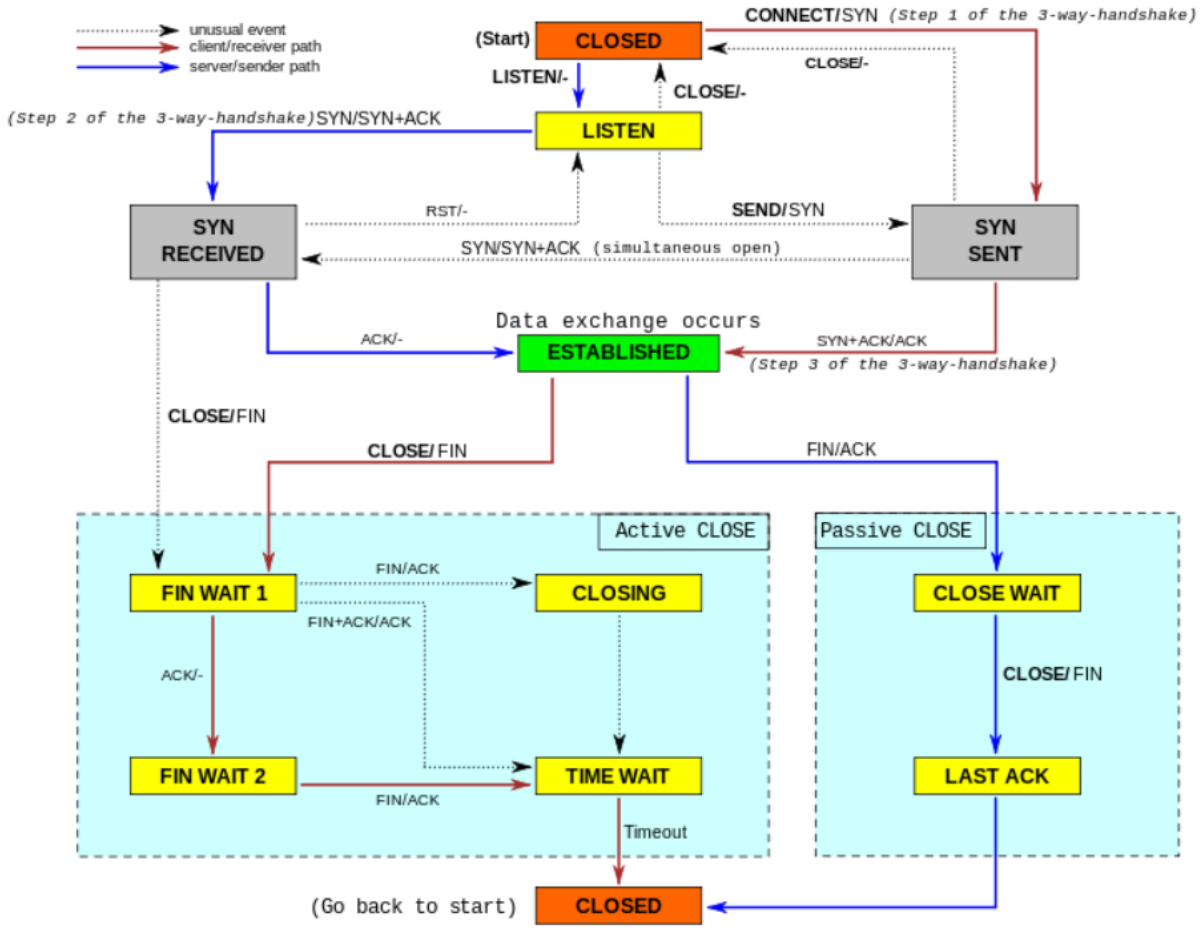
Přenos dat - established

Je to stav „etablished“. Dle režimu je více variant.

Ukončení spojení

Ukončení spojení je realizováno dle situace

Obrázek popisuje přehledně celou stavovou část TCP protokolu.



TCP hlavička

Bity	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
0	zdrojový port																cílový port															
32	číslo sekvence																															
64	potvrzený bajt																															
96	offset dat				rezervováno						příznaky						okénko															
128	kontrolní součet																Urgent Pointer															
160	volby (volitelné)																															
192	volby (pokračování)																								výplň (do 32)							
224	data																															

Data Offset - specifikuje číslo vyjádřené 32bitovým slovem. Indikuje, kde data v segmentu začínají - data přenášená tímto datagramem.

Urgent Pointer - údaj je platný pouze, pokud je nastaven příznak URG

Kontrolní součet – hlavičky, dat a pseudohlavičky (IPadresy z IP paketu)

Alternativy TCP

Problémy TCP

- **blokování čela fronty**, kvůli kterému aplikace po ztrátě jednoho paketu nedostává následující pakety do té doby, dokud není ztracený paket znovu poslán a úspěšně přijat. To způsobuje **problémy realtimovým aplikacím jako streamovaná média (např. internetové rádio), realtimové hry pro více hráčů a VoIP.**

- **Složitost TCP může být problém pro vestavěná zařízení (anglicky *embedded systems*).** Například bootování po síti, které používá TFTP (PXE). Dále přenos dat mezi dvěma uzly, které jsou oba za NATem je jednodušší, když vám v cestě nestojí složitý protokol jako TCP.

- **TCP má problémy v prostředí s velkou šířkou pásma**. TCP algoritmy pro zamezení zahlcení (řízení toku) fungují velmi dobře pro ad-hoc prostředí, kde odesílatel dat není předem znám.

UDP - poskytuje aplikaci kontrolu/ovládání nad multiplexováním a ověřováním kontrolních součtů, ale UDP neprovádí fragmentaci proudu dat do paketů a zpátky jejich rekonstruování, ani opětovné posílání ztracených paketů. **Vývojář aplikace si musí napsat uvedené funkce tak, jak vyhovuje jeho potřebám**, nebo je nahradit použitím samoopravných kódů (anglicky forward error correction) nebo interpolace.

SCTP (Stream Control Transmission Protocol)- je transportní protokol nad IP, který **poskytuje spolehlivé, datagramové služby nepříliš odlišné od TCP.** Je **novější a mnohem složitější** než TCP. **Používá se v telekomunikačních sítích pro přenos signalizace,** ale v běžném provozu se nedočkal širokého nasazení, ačkoliv je obzvláště navržený k tomu pro použití v situacích, kdy jsou spolehlivost a téměř real-time ohledy důležité.

VTP (Venturi Transport Protocol) (VTP) - patentovaný **proprietární protokol**, který je navržen tak, aby nahradil TCP a transparentně **překonal vnímané nedostatky vztahující se k bezdrátovému přenosu dat. V** případě, že je prostředí dopředu známo, mohou časové principy protokolu, jako je například asynchronní typ přenosu snížit režii pro znovu odeslání dat.

UDT (User Data Transfer) založené na User Datagram Protocol, **má v sítích, které mají vysoký rozsah latence, lepší účinnost a „férovost“ než TCP.**

Víceúčelový Transakční Protokol (MTP/IP) je patentován jako **proprietární software**, který je navržen tak, aby adaptivně dosahoval vysoké propustnosti a výkonu v nejrůznějších síťových podmínkách, a to zejména tam, kde TCP je vnímán jako nevyhovující nebo nedostatečný.

QUIC (*Quick UDP Internet Connections*) je experimentální protokol využívající UDP, který vyvinula společnost Google pro novou verzi protokolu HTTP.

Zdroj wikipedie