Transportna plast

Naloge transportne plasti so:

- povezovanje dveh oddaljenih *procesov*
- multipleksiranje/demultipleksiranje komunikacije med procesi
- zanesljiv prenos podatkov
- kontrola pretoka in zasičenja

Vrata - port

FTP-data: 20, FTP-cmd: 21, SSH: 22, Telnet: 23, SMTP: 25, DNS: 53, HTTP: 80, POP3: 110, IMAP: 143, IRC: 194, HTTPS: 443

User Datagram Protocol: UDP

Nepovezavna storitev, nudi le best-effort: ni zagotavljanja vrstnega reda, ni nadzora zamašitev, ...

Dolžina glave je le 8B.

16b	16b
Izvorna vrata	*Ponorna vrata*
Dolžina (skupaj z glavo) v B	*Internetna kontrolna vsota*

Internetna kontrolna vsota

Pošiljatelj: pošiljatelj sešteje 16 bitne besede in shrani eniški komplement = kontrolna vsota
Prejemnik: sešteje 16 bitne besede skupaj s kontrolno vsoto → dobiti mora same enice

Potrjevanje

sprotno potrjevanje: po vsakem pošiljanju počakaj na potrditev

tekoče pošiljanje: večji razpon za številčenje paketov, shranjevanje paketov na obeh straneh

- ponavljanje N potrjenih (*go-back-N*)
  - pošiljatelj hrani okno največ dovoljenih nepotrjenih paketov
  - ACK(n) potrdi vse pakete do vključno n
  - časovna kontrola za najstarejši paket, ko poteče pošlji vse nepotrjene pakete na novo
- ponavljanje izbranih (*selective repeat*)
  - prejemnik shranjuje sporočila in jih sortira pred dostavo
  - pošiljatel ponovno pošlje le tiste pakete za katere ni prejel potrdila
  - vsak paket ima svojo časovno kontrolo

neposredno: uporaba ACK in NCK **posredno**: samo ACK, namesto NCK se ponovi ACK zadnjega segmenta

Transfer Control Protocol: TCP

Povezavna storitev, full duplex, zanesljiv, kontrola pretoka, kontrola zasičenja, *tekoče pošiljanje in neposredno potrjevanje*

16b			16b
Izvorna vrata			*Ponorna vrata*
Dolžina (skupaj z glavo) v B			*Internetna kontrolna vsota*
zaporedna št. (B)			
št. potrditve (B)			
Dolzina glave	*0*	*zastavice*	*sprejemno okno*
Internetna kontrolna vsota			*karalec urg. data*
opcije (spremenljiva dolžina)			

Dolžina glave (4b) je podana v številu 32-bitnih besede

Sprejemno okno (16b) št. bajtov, ki jih sprejemnik lahko sprejme

Zastavice (9b):

- URG*: urgentni podatki
- ACK: potrditev povezave
- PSH: takoj predaj aplikaciji
- RST, SYN, FIN: vzpostavljanje in rušenje povezave

Vzpostavitev povezave - trojno rokovanje

1. Odjemalec pošlje segment z zastavico `SYN` (sporoči začetno številko segmenta, ni podatkov)
2. Strežnik vrne segment `SYN ACK` (rezervira medpomnilnik, odgovori z začetno številko svojega segmenta)
3. Odjemalec vrne `ACK`, lahko že s podatki ("štuporama")

Rušenje povezave

1. odjemalec pošlje segment `TCP FIN` strežniku
2. strežnik potrdi z `ACK`, zapre povezavo, pošlje `FIN`
3. odjemalec prejme strežnikov `FIN`, potrdi ga z `ACK` (počaka časovni interval, da po potrebi ponovno pošlje `ACK`, če se ta izgubi)
4. strežnik sprejme `ACK`, končano

Nastavitev časovne kontrole

Na potrditev moramo čakati vsaj `RTT` (Round Trip Time).

$$\texttt{OcenRTT}[i] = (1-\alpha)\texttt{OcenRTT}[i-1] + \alpha\texttt{IzmerRTT}[i]$$
$$\texttt{DevRTT}[i] = (1-\beta)\texttt{DevRTT}[i-1] + \beta\left|\texttt{IzmerRTT}[i] - \texttt{OcenRTT}[i]\right|$$
$$\texttt{ČakalniInterval}[i] = \texttt{OcenRTT}[i] + 4\texttt{DevRTT}[i]$$

Način potrjevanja

TCP uporablja podobno strategijo kot *ponavljanje N nepotrjenih*. Pošiljatelj ima časovno kontrolo le za najstarejši nepotrjen segment, vendar ob poteku časovne kontrole ne pošlje vseh segmentov v oknu, temveč le najstarejši nepotrjen segment.

RFC2018 vpeljuje potrjevanje le izbranih paketov.

Dogodek pri prejemniku	*Odziv prejemnika*
Sprejem segmenta s pričakovano številko, vsi prejšnji že potrjeni.	Počakaj na naslednji segment max 500 ms. Če ta pride v tem intervalu, izvedi zakasnjeno potrditev obeh (*delayed ACK*). Če ne pride v tem intervalu, potrdi samo prejetega.
Isto kot zgoraj, a potrditev za prejšnji segment še ni bila poslana.	Takoj pošlji *kumulativno potrditev* za oba segmenta brez izvajanja zakasnjene potrditve.
Sprejem segmenta s previsoko številko (*zaznamo vrzel*)	Takoj potrdi zadnji še sprejeti segment (*pošlji duplikat ACK*).
Sprejem segmenta z najnižjo številko iz vrzeli (*polnjenje vrzeli*)	Takoj potrdi segment.

Hitro ponovno pošiljanje (fast retransmit)

Ponovno pošiljanje se običajno izvede po preteku časovne kontrole. Če pa pošiljatel prejme 3 potrditve za že potrjen paket, takoj izvede ponovno pošiljanje.

Kontrola pretoka

Usklajevanje hitrosti pošiljanja med pošiljateljem in prejemnikom.

Prejemnik sporoča razpoložljiv prostor v pomnilniku v polju 'sprejemno okno':

$$\texttt{rwnd} = \texttt{velikost} - [\texttt{lastByteRcvd} - \texttt{lastByteRead}]$$

Oznake

`rwnd`	...	recieve window
`cwnd`	...	congestion window
`MSS`	...	maximum segment size
`RTT`	...	round trip time
`ssthresh`	...	slow start threshold

Nadzor zasičenja

min(`rwnd`, `cwnd`) določa največ nepotrjenih B preden moramo ustaviti pošiljanje.

TCP Tahoe: osnovna verzija; 2 fazi: *počasen začetek* in *izogibanje zasičenju*; po izgubi paketa `cwnd` = 1 `MSS`.

TCP Reno: 3 faze: *počasen začetek*, *izogibanje zasičenju* in *hitra obnova*; če dobiš 3 kopije ACK že potrjenih podatkov, `cwnd` = `cwnd`/2 + 3 `MSS`.

TCP Vegas: dodano zaznavanje sitacij, ki vodijo v zasičenje in *linearno zmanjševanje hitrosti* ob zasičenju.

Na začetku: `cwnd` = 1`MSS`; faza počasnega začetka; `ssthresh` = ∞.

- **Faza počasnega začetka:**
  Za vsak prejet ACK: `cwnd` += min(`N`, `MSS`), kjer je `N` št. B, ki jih je potrdil ta ACK.
  Alt. rešitev: Za vsak prejet ACK: `cwnd` += `MSS`.
  Na ta način se skozi čas `cwnd` povečuje eksponentno.
  - Če je `cwnd` $\geq$ `ssthresh`, preidemo v fazo izogibanja zasičenju.
  - Če prejmemo 3 ACK, že potrjenega segmenta, gremo v *fazo hitre obnove* `cwnd` = `ssthresh` + 3`MSS`, `ssthresh` = `cwnd` / 2
- **Faza izogibanja zasičenju:**
  Števemo B, ki jih potrdijo ACK. Ko to število preseže `cwnd`, resetiramo števec in `cwnd` += MSS.
  Alt. rešitev: Za vsak ACK, ki potrdi nove podatke povečamo `cwnd`+ = `MSS` · `MSS`/`cwnd`
  Na ta način se skozi čas `cwnd` povečuje linearno.
  - Če poteče časovna pontrola za najstarejši segment, gremo v *fazo počasnega začetka* in `cwnd` = MSS, `ssthresh` = `cwnd` / 2.
  - Če prejmemo 3 ACK, že potrjenega segmenta, gremo v *fazo hitre obnove* `cwnd` = `ssthresh` + 3`MSS`, `ssthresh` = `cwnd` / 2
- **Faza hitre obnove:**
  Za vsak podvojeni ACK: `cwnd` += `MSS`.
  - Če dobimo ACK, ki potrjuje nove podatke, gremo v *fazo izogibanja zasičenju* in `cwnd` += `ssthresh`.
  - Če preteče nek timeout, gremo v *fazo počasnega začetka* in `cwnd` = MSS, `ssthresh` = `cwnd` / 2.

TCP konvergira k pravični delitvi pasovne širine med uporabniki.

Aplikacijska plast

HTTP

Glava

```
Metoda URL Verzija
Ime polja 1: vrednost 1
...
Ime polja n: vrednost n
[prazna vrstica]
TELO
```

Metode

`GET`	zahteva objekta
`POST`	zahteva objekta + poslane vrednosti (obrazci)
`HEAD`	zahteva, na katero strežnik odgovori z odgovorom brez zahtevanega objekta (uporabno za razhroščevanje)
`PUT`	(HTTP 1.1) – nalaganje na strežnik (upload)
`DELETE`	(HTTP 1.1) – brisanje s strežnika
`TRACE`	razhroščevanje (echo-odmev zahtevka, podobno PING)
`CONNECT`	povezava preko med-strežnika
`OPTIONS`	povpraševanje o možnih opcijah pri zahtevku

Statusi

- **1xx: informativne kode**
- **2xx: uspešno**
  200: OK

- **3xx: preusmeritev**
  301: Moved Permanently – prestavljen dokument
- **4xx: napake pri odjemalcu**
  400: Bad Request – sintaksa
  404: Not Found – ni dokumenta
- **5xx: napake na strežniku**
  500: Internal Server Error
  505: HTTP Version Not Supported

Vrste HTTP povezav

1. **nevztrajne (nonpersistent)**: za vsak prenašani objekt (stran, sliko) se vzpostavi nova TCP povezava
   2 RTT/objekt (rokovanje + prenos)
2. **vztrajne (persistent)**: strežnik uporabi isto povezavo za pošiljanje več objektov, strežnik pusti povezavo odprto
   1 RTT/objekt (prenos)
3. **vztrajne s cevovodom (persistent, pipelined)**: tekoče pošiljanje več zahtev naenkrat, brezčakanja na prejem prejšnjih

Medstrežnik

Pogojni zahtevek: `If-modified-since: <datum>`.

Strežnik pošlje nov stran ali pa `HTTP/1.1 304 Not Modified`

File Transfer Protocol - FTP

Dve ločeni TCP povezavi:

- vrata TCP 21 (kontrolna povezava): ukazi za prenos datotek, uporabniško ime/geslo, menjava map, ...
- vrata TCP 20 (prenos podatkov) na zahtevo odjemalca strežnik odpre povezavo, po kateri prenaša podatke

Aktivni in pasivni način

Aktivni: odjemalec se iz naključnega porta X poveže na strežnik prek porta 21 (kontrolna povezava), nato se strežnik poveže na odjemalca iz porta 20 na naključen port Y.

Pasivni: odjemalec se iz naključnega porta X poveže na strežnik prek porta 21 (kontrolna povezava). Odlemalec pošlje ukaz PASV. Strežnik odgovori z naključnim portom Y, ki ga je odpru za podatkovno povezavo. Odjemalec se nato iz naključnega porta Z poveže na Y.

Ukazi

`USER <ime> PASS <geslo> LIST RETR <ime datoteke> STOR <ime datoteke>`

Odgovori

- `331` Username OK, password required
- `125` Data connection open, transfer starting
- `452` Error writing file
- `425` Can't open data connection

Elektronska pošta

Sporočilo je sestavljeno iz glave, prazne vrstice in telesa.

Komunikacija med poštnimi strežniki: **SMTP**

Dostop do pošte:

- **POP**: preprost, ne podpira urejanja pošte na strežniku (le lokalno) zato ni primeren za dostop do istega predala iz več naprav.
  Ukazi: `user <ime>, pass <geslo>, list, retr <id>, dele <id>, quit`
- **IMAP**: omogoča urejanje sporočil po mapah na strežniku, možen prenos le dela sporočila
- **HTTP**: prek brskalnika (recimo Gmail)

Domain Name System - DNS

Hierarhična organizacija strežnikov

- **Korenski strežniki**: 13 strežnikov (A-M), vsak je replicirana gruča
- **Top Level Domain - TLD** strežniki:
  generične domene: 7 prvotnih (com, edu, gov, mil, org, net, biz); ∼1500 dodatnih (info, blog, ...); ∼255 domen za države (si, it, de, tv, am, gl, ...)
- **Avtoritativni strežniki**: organizacija z javnimi računalniki (UL: uni-lj)

Iterativna poizvedba:

- strežnik vrne bodisi končni odgovor ali pa naziv strežnika za naslednje povpraševanje (primer: lokalni strežnik iterativno povpraša ostale)

Rekurzivna poizvedba:

- strežnik poišče preslikavo imena in vrne odgovor (primer: naša poizvedba lokalnemu strežniku)
- razbremenimo končne kliente komunikacije in povpraševanja
- možnost centralnega predpomnenja v lokalnem strežniku!

DNS record

`(Name, Value, Type, TTL)`

Time To Live - TLL: čas veljavnosti zapisa

Type:

- **A/AAAA** *(addres)*
  Name: ime računalnika (poddomena), Value: IPv4/IPv6 naslov
- **NS** *(name server)*
  Name: ime domene, Value: ime avtoritativenga DNS strežnika
- **CNAME** *(cannonical name)*
  Name: alias ime, Value: pravo (kanonično) ime
- **MX** *(mail exchange)*
  Name: alias poštnega strežnika, Value: pravo ime poštnega strežnika

P2P storitve

BitTorrent

strategija: sosede vpraša po razpoložljivih koščkih datotek in zahteva najprej tiste manjkajoče, ki so najbolj redki med sosedi *(rarest first!)*

pravičnost: opazujemo hitrost prejemanja od sosedov in jim pošiljamo koščke s sorazmerno visoko hitrostjo

pomembna je **vzpodbuda za sodelovanje**

Skype

NAT povzroča težave v P2P arhitekturah, ker zunanji odjemalci ne morejo direktno kontaktirati odjemalca za prehodom NAT.

rešitev:

- odjemalce A vzpostavi z B zvezo preko nadzornih vozlišč NA in NB,
- NA in NB izbereta tretje premostitveno (relay) nadzorno vozlišče (NR), s katerim A in B vzpostavita sejo
- premostitveno vozlišče poskuša zagotoviti neposredno povezavo med odjemalcema

Sejna plast

Naloge:

- vzpostavljanje, rušenje, vzdrževanje sej med aplikacijama (potek dialoga med aplikacijama)
- odgovornost za obnovitvene točke (checkpoints) seje in obnovo (recovery), če seja ne uspe
- sinhronizacija podatkov iz različnih tokov in virov

V eni seji je lahko več transportnih povezav in ena transportna povezava lahko sega čez več sej.

Predstavitvena plast

Storitve

- **predstavitev podatkov** *(binarni tipi)*: različni sistemi predstavljajo binarne tipe na različen način; uporaba sintakse **ASN.1** (Abstract Syntax Notation 1) zmanjša število vseh možnih preslikav
- **predstavitev alfanumeričnih znakov** *(združljivost kodnih strani)*: znaki so predstavljeni s številkami po kodnem sistemu, potrebno zagotoviti, da se prenašajo pravi znaki
- **stiskanje podatkov**: omogoča zmanjšanje velikosti podatkov in s tem pohitritev prenosa (jpeg, mpeg)
- **zaščita podatkov** *(kriptiranje)*: varnostni mehanizem, ki omogoča vpeljavo zaupnosti v komunikacijo

Kriptogorafija

m	...	čistopis
$K_A(m)$	...	kriptogram, kriptiran s ključem A
$K_B(K_A(m))$	...	odkriptiran z dekripcijskim ključem B

Metode kriptiranja

Glede na način kriptiranja (*algoritem*):

- **substitucija** (*zamenjava znakov z drugimi*)
  - **Cezarjev kriptogram**: vsako črko zamenjamo z naslednjo k-to po abecedi.
  - **Vigenèr-jev kriptogram**: Ključ K je neka beseda dolžine n. Vsako črko sporočila M zamenjamo: $M[i] = M[i] + K[i\%n]$
    Računamo v grupi seštevanja ostankov črk.
  - **Porterjev kriptogram**: kriptiramo po 2 zanak skupaj (z veliko abecedo)
- **transpozicija** (*zamenjava vrstnega reda zankov*)
- **kombinirane metode**

Glede na velikost sporočila:

- **znakovna**
- **bločna**

Glede na ključe:

Simetrično enkripcija: $K_S(K_S(m)) = m$.

Asimetrična enkripcija (z javnim in zasebnim ključem): $K_J(K_Z(m)) = m = K_Z(K_J(m))$

Bločna kriptografija

Permutacijska škatla s ključem $(abc\dots)$ slika a. bit na 0. mesto, b. bit na 1. mesto, itd.

Substitucijska škatla: dekoder (slika števila v prižgan *en* bit) → s-škatla → koder (slika *en* prižgan bit v št.)

Data Encryption Standard - DES

64b blok → transpozicija T_1 → iteracija 1 → $\cdots$ → iteracija 16 → zamenjamo prvo in zadnjo polovico bitov → transpozicija T_1^{-1} → 64b kriptogram

izhod iteracije $i-1$ razdelimo na polovici L_{i-1} in R_{i-1} → $L_i = R_{i-1}$ in $R_i = L_{i-1} \otimes f(R_{i-1}, K_i)$

Trojni DES

Sporočilo se kriptira (K_1), dekriptira (K_2), kriptira (K_3).

Združljiv z DES, če $K_2 = K_3$.

Advanced Encryption Standard - AES

Najbolj učinkovita in varna metoda.

Bloki velikosti 128b, ključi pa 128/192/256b.

Verižno kriptiranje

Pošiljatelj s prvom sporočilom pošlje inicializacijski vektor $c(0)$.

Naslednja sporočila kriptira: $c(i) = K_S(m(i) \otimes c(i-1))$

Prejemnik sporočila odkriptira:
$m(i) = K_S^{-1}(c(i)) \otimes c(i-1)$

Asimetrična kriptografija

RSA:

- izberemo veliki praštevili p, q, izračunamo:
  $$n = pq \qquad z = (p-1)(q-1)$$
- izberemo e tako, da $\gcd(e, z) = 1$
- izberemo d tako, da $ed \equiv_z 1$
- javni ključ: (n, e), zasebni ključ (n, d)

$$\text{kriptiranje} : m \mapsto c = m^e \bmod n$$

$$\text{dekriptiranje} : c \mapsto m = c^d \bmod n$$