# **6. Protokoly transportní vrstvy**

## **Účel, funkce transportní vrstvy, protokoly transportní vrstvy**

### **Účel transportní vrstvy**

* Transportní vrstva slouží k logické komunikaci mezi aplikacemi běžícími na různých koncových zařízeních
* Propojuje aplikační vrstvu s nižšími vrstvami, které jsou zodpovědné za síťovou komunikaci
* Transportní vrstva zahrnuje dva protokoly:
  + Transmission Control Protocol (TCP)
  + User Datagram Protocol (UDP)

### **Funkce transportní vrstvy**

#### **1. Sledování jednotlivých konverzací**

* Na transportní vrstvě se komunikace mezi zdrojovou a cílovou aplikací nazývá konverzace
* Transportní vrstva udržuje a sleduje tyto konverzace mezi více aplikacemi (spíše u TCPka)

#### **2. Segmentace dat a sestavování segmentů**

* Přijímá data z aplikační vrstvy, dělí je na menší segmenty vhodné pro další enkapsulaci v síťové vrstvě
* Sítě mají omezení na velikost dat, kolik může jeden paket obsahovat
* Transportní vrstva odpovídá za rozdělení dat na vhodně velké bloky
* Podle použitého protokolu se tyto bloky nazývají:
  + Segmenty – u TCP protokolu
  + Datagramy – u UDP protokolu

#### **3. Přidání hlavičky s informacemi**

* Do hlavičky každého segmentu se přidají informace potřebné pro správu přenosu dat
* Například informace v hlavičce používá přijímající hostitel k rekonstruování celkové komunikace (v toku) ze segmentů čí datagramů

#### **4. Identifikace aplikací**

* K předání dat správné aplikaci, transportní vrstva identifikuje cílovou aplikaci číslem portu
* Každému procesu, který potřebuje přistupovat k síti, je přiřazeno číslo portu, které je jedinečné pro daného hostitele

#### **5. Multiplexing**

#### transportní vrstva dokáže provozovat několik spojení najednou

#### pro rozlišení spojení používá tzv. porty

**6. Error control**

* pouze u tcp
* zajišťuje, aby data byla přenesena spolehlivě
* segmenty mají sekvenční čísla, které slouží k sestavení dat ve správném pořadí
* pomocí checksum pole lze poznat poškozený segment
* potvrzuje se přijetí dat
* v případě potřeby se žádá a poslání znovu

**7. Flow control**

* hlídá rate, kterým jsou data posílána, aby nedošlo k přetížení sítě (pouze TCP)

### **Protokoly transportní vrstvy**

#### **TCP (Transmission Control Protocol)**

##### poskytuje spolehlivý přenos správně seřazených a nenarušených dat

##### používá kontrolní mechanismy jako checksumy a ACKs, když je detekován error, data jsou poslána znovu

##### implementuje **flow control** viz. funkce transportní vrstvy

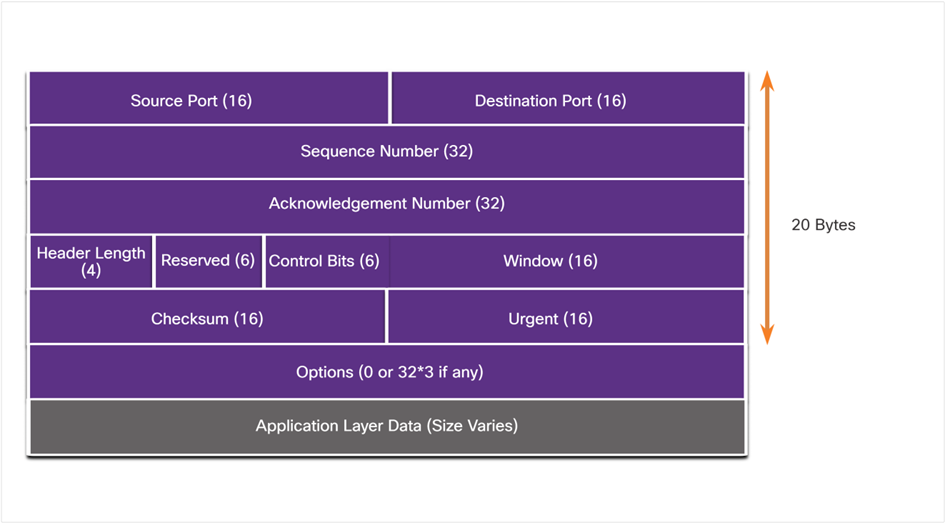
##### sleduje segmenty, potvrzuje přijetí dat, seřazuje data, která mohla přijít ve špatném pořadí

##### tzv. **connection-oriented protocol** – než posílá, ustálí s cílem spojení

##### tcp je používán v protokolech ftp, http, smtp, či ssh

##### struktura tcp Záhlaví

##### **TCP hlavička**



* Source Port – 16-bitové pole, které identifikuje zdrojovou aplikaci podle čísla portu
* Destination Port – 16-bitové pole, které identifikuje cílovou aplikaci podle čísla portu
* Sequence Number – 32-bitové pole určené pro sestavování dat
* Acknowledgement Number – 32-bitové pole označující, že data byla přijata a je očekáván další byte ze zdroje
* Header Length – 4-bitové pole, které udává délku hlavičky TCP segmentu
* Reserved – 6-bitové pole rezervované pro budoucí použití
* Control Bits– 6-bitové pole označující účel a funkci segmentu
* Window Size – 16-bitové pole, které udává počet bajtů, které lze přijmout najednou
* Checksum – 16-bitové pole sloužící ke kontrole chyb v hlavičce a datech segmentu
* Urgent – 16-bitové pole, které označuje zda jsou obsažená data naléhavá

##### **Použití**

* Databáze, webové prohlížeče, emailoví klienti
* Přenos uložených videí a zvukových nahrávek

#### 

#### **UDP (User Datagram Protocol)**

##### jednoduchý a **rychlý** protokol, který negarantuje spolehlivé doručení

##### **neprovádí flow control**

##### **neprovádí error checking**

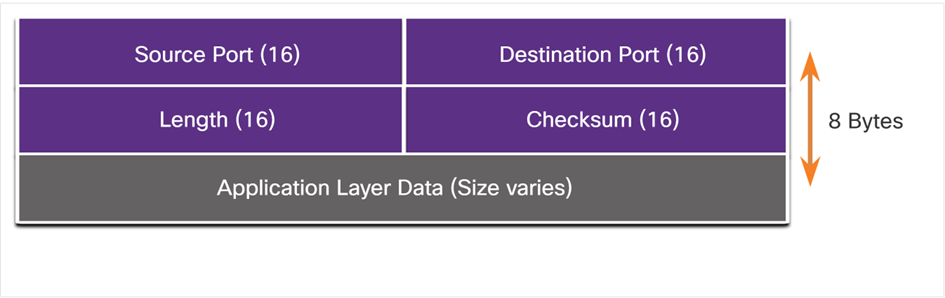
##### **neudržuje spojení** s cílem

##### vhodný pro aplikace s potřebou **velmi rychlé komunikace** bez důrazu na spolehlivost

##### používáno v dns, dhcp, tftp, ntp, VoIP, online hry atd.

##### existují **nadstavbové protokoly**, které využívají udp a **implementují** funkce jako **error checking, či flow control**, např: RTP (real time transport control) sloužící pro real time streamování audia nebo videa

##### **UDP hlavička**



* Source Port – 16-bitové pole, které identifikuje zdrojovou aplikaci podle čísla portu
* Destination Port – 16-bitové pole, které identifikuje cílovou aplikaci podle čísla portu
* Length– 16-bitové pole, které udává délku hlavičky datagramu
* Checksum – 16-bitové pole sloužící ke kontrole chyb v hlavičce a datech datagramu

##### **Použití**

* Streamování videí
* Jednoduché aplikace pro žádost a odpověď (DNS, DHCP)
* Aplikace, které se o spolehlivost starají sami (SNMP, TFTP)

## **Porty aplikací**

* TCP/IP porty jsou **virtuální kanály** umožňující více aplikacím komunikovat na síti
* existuje 65535 možných portů
* prvních 1024 portů (0-1023) se nazývá **well-known porty**, jsou reservovány pro specifické aplikace, jako http(80), ftp(20,21), https(443)
* 1024 - 49151 jsou registrované porty používané aplikacemi registrovanými u IANA (Internet Assigned Numbers Authority)
* 49152-65535 jsou **efemerální nebo také privátní** porty, může je používat jakákoli aplikace pro navázání spojení

## **3-way handshake a ukončení spojení**

* Three-way handshake proces slouží k navázání spojení u TCP

Má za úkol:

* Zajistit dostupnost
* Aktivitu služby
* Informuje příjemce o připojení a vytvoření sessionu/relace

### **Navázání spojení**

1. **SYN** – klient si vyžádá komunikační relaci client-to-server (Obsahuje SEQ, který se nastaví na náhodné číslo → Vytvořeno clientem)
2. **ACK a SYN** – server potvrdí komunikační relaci client-to-server a vyžádá si komunikační relaci server-to-client (Obsahuje ACK, který se nastaví na náhodné číslo → Vytvořeno serverem)
3. **ACK** – klient potvrdí komunikační relaci server-to-client

### **Ukončení spojení**

1. **FIN** – když už klient nemá další data k odeslání, odešle ukončovací segment
2. **ACK** – server potvrdí přijetí zprávy FIN a ukončí relaci client-to-server
3. **FIN** – server odešle klientovi zprávu FIN, aby ukončil relaci server-to-client
4. **ACK** – client potvrdí přijetí FIN zprávy od serveru a ukončí relaci

**Bezpečnostní hrozba / útok — SYN flood**

* Forma DoS (Denial of Service)
* Spočívá v tom, že útočník vytvoří mnoho připojení se serverem, tím, že posílá SYN requesty
* Server na všechny musí odpovědět pomocí SYN ACK
  + Segmenty můžou obsahovat falešnou IP, takže útočník nedosatane žádnou odpověď
  + Může si vesele dál poslat SYN requesty
* Server se zahltí tím, že má spoustu otevřených připojení, na které čeká, než se dokončí — Což se nikdy nedokončí