

Učební texty k státní bakalářské zkoušce
Programování
Sítě a internetové technologie

študenti MFF

15. augusta 2008

6 Sítě a internetové technologie

Požadavky

- Architektura ISO/OSI
- Rodina protokolu TCP/IP (ARP, IPv4, IPv6, ICMP, UDP, TCP) - adresace, routing, fragmentace, spolehlivost, flow control, congestion control, NAT
- Rozhraní BSD sockets
- Spolehlivost - spojované a nespojované protokoly, typy, detekce a oprava chyb
- Bezpečnost - IPSec, principy fungování AH, ESP, transport mode, tunnel mode, firewalls
- Internetové a intranetové protokoly a technologie - DNS, SMTP, FTP, HTTP, NFS, HTML, XML, XSLT a jejich použití.

6.1 Architektura ISO/OSI

Úvod

Definice

Síťový model je ucelená představa o tom, jak mají být sítě řešeny (obsahuje: počet vrstev, co má která vrstva na starosti; neobsahuje: konkrétní představu jak která vrstva plní své úkoly - tedy konkrétní protokoly). Příkladem je *referenční model ISO/OSI* (konkrétní protokoly vznikaly samostatně a dodatečně). **Síťová architektura** navíc obsahuje konkrétní protokoly - napr. *rodina protokolů TCP/IP*.

Referenčný model ISO/OSI (International Standards Organization / Open Systems Interconnection) bol pokusom vytvoriť univerzálnu sieťovú architektúru - ale skončil ako sieťový model (bez protokolov). Pochádza zo „sveta spojov“ - organizácie ISO, a bol „oficiálnym riešením“, presadzovaným „orgánmi štátu“; dnes už prakticky odpísaný - prehral v súboji s TCP/IP. ISO/OSI bol reakciou na vznik proprietárnych a uzavretých sietí. Pôvodne mal model popisovať chovanie otvorených systémov vo vnútri aj medzi sebou, ale bolo od toho upustené a nakoniec z modelu ostal len sieťový model (popis funkcionality vrstiev) a konkrétne protokoly pre RM ISO/OSI boli vyvíjané samostatne (a dodatočne zaraďované do rámca ISO/OSI).

Model vznikol maximalistickým spôsobom - obsahoval všetko čo by mohlo byť v budúcnosti potrebné. Vďaka rozsiahlosti štandardu sa implementovali len jeho niektoré podmnožiny - ktoré neboli (vždy) kompatibilné. Vznikol GOSIP (Government OSI Profile) určujúci podmnožinu modelu, ktorú malo mať implementované všetko štátne sieťové vybavenie. Naproti tomu všetkému TCP/IP vzniklo naopak - najprv navrhnutím jednoduchého riešenia, potom postupným obohacovaním o nové vlastnosti (tie boli zahrnuté až po preukázaní „životaschopnosti“).

7 vrstev

Kritériá pri návrhu vrstiev boli napr.: rovnomerná vyťaženosť vrstiev, čo najmenšie dátové toky medzi vrstvami, možnosť prevziať už existujúce štandardy (X.25), od-

lišné funkcie mali patriť do odlišných vrstiev, funkcie na rovnakom stupni abstrakcie mali patriť do rovnakej vrstvy. Niektoré vrstvy z finálneho návrhu sa používajú málo (relačná a prezentačná), niektoré zase príliš (linková - rozpadla sa na 2 podvrstvy LLC+MAC).

aplikačná vrstva prezentačná vrstva relačná vrstva	vrstvy orientované na podporu aplikácií
transportná vrstva	prispôsobovací vrstva
sieťová vrstva linková vrstva fyzická vrstva	vrstvy orientované na prenos dát

Fyzická vrstva sa zaoberá prenosom bitov (kódovanie, modulácia, synchronizácia...) a ponúka teda služby typu pošli a príjmi bit (pričom neinterpretuje význam týchto dát). Pracuje sa tu s veličinami ako je *šírka pásma, modulačná a prenosová rýchlosť*.

Linková vrstva prenáša vždy celé bloky dát (rámce/frames), používa pritom fyzickú vrstvu a prenos vždy funguje len k priamym susedom. Môže pracovať spoľahlivo či nespoľahlivo, prípadne poskytovať QoS/best effort. Ďalej zabezpečuje riadenie toku - zaistenie toho, aby vysielajúci nezahltil príjemcu. Delí sa na dve podvrstvy - MAC (prístup k zdieľanému médiu - rieši konflikty pri viacnásobnom prístupe k médiu) a LLC (ostatné úlohy).

Sieťová vrstva prenáša pakety (packets) - fakticky ich vkladá do linkových rámcov. Zaručuje doručenie paketov až ku konečnému adresátovi (tj. zabezpečuje smerovanie). Môže používať rôzne algoritmy smerovania - ne/adaptívne, izolované, distribuované, centralizované... (v architektúre TCP/IP je to IP vrstva)

Transportná vrstva zabezpečuje komunikáciu medzi koncovými účastníkmi (end-to-end) a môže meniť nespoľahlivý charakter komunikácie na spoľahlivý, menej spoľahlivý na viac spoľahlivý, nespojovaný prenos na spojovaný... Príkladom sú napr. TCP a UDP. Ďalšou úlohou je rozlišovanie jednotlivých entít (na rozdiel od napr. sieťovej vrstvy) v rámci uzlov - procesy, démony, úlohy (rozlišuje sa zväčša nepriamo - napr. v TCP/IP pomocou portov).

Relačná vrstva zaisťuje vedenie relácií - šifrovanie, synchronizáciu, podporu transakcií. Je to najkritizovanejšia vrstva v ISO/OSI modeli, v TCP/IP úplne chýba.

Prezentačná vrstva slúži na konverziu dát, aby obe strany interpretovali dáta rovnako (napr. reálne čísla, rôzne kódovanie textov). Ďalej má na starosti konverziu dát do formátu, ktorý je možné preniesť: napr. linearizácia viacrozmerných polí, dátových štruktúr; konverzia viacbajtových položiek na jednotlivé byty (little vs. big endian). *Poznámka:* Zápis čísla 1234H v Big endian je [12:34::-] (sun, motorola), v Little endian [-::-34:12] (intel, amd, ethernet).

Aplikačná vrstva mala pôvodne obsahovať aplikácie - ale tých je veľa a nebolo možné ich štandardizovať. Teraz teda obsahuje len „jadro“ aplikácií - tie, ktoré malo zmysel štandardizovať (email a pod.). Ostatné časti aplikácií (GUI) boli vysunuté nad aplikačnú vrstvu.

Kritika

Model ISO/OSI:

- je príliš zložitý, ťažkopádny a obtiažne implementovateľný
- je príliš maximalistický
- nerešpektuje požiadavky a realitu bežnej praxe
- počítal skôr s rozľahlými sieťami ako s lokálnymi
- niektoré činnosti (funkcie) zbytočne opakuje na každej vrstve
- jednoznačne uprednostňuje spoľahlivé a spojované prenosové služby (ale tie sú spojené s veľkou réžiou ⇒ spoľahlivosť si efektívnejšie zabezpečia koncové uzly)

Možnosť nespoľahlivého/nespojovaného spojenia bolo pridané do štandardu až dodatočne, napriek tomu bol porazený architektúrou TCP/IP. Používajú sa však niektoré prevzaté prokoly - X.400 (elektronická pošta), X.500 (adresárové služby - odľahčením vznikol úspešný protokol LDAP).

6.2 Rodina protokolů TCP/IP (ARP, IPv4, IPv6, ICMP, UDP, TCP) – adresace, routing, fragmentace, spolehlivost, flow control, congestion control, NAT

ISO/OSI	TCP/IP
aplikační vrstva prezentační vrstva relační vrstva	aplikační vrstva
transportní vrstva	transportní vrstva
síťová vrstva	síťová vrstva (též IP vrstva)
linková vrstva fyzická vrstva	vrstva síťového rozhraní

Obvyklé označenie je *TCP/IP protocol suite* (súčasťou je viac ako 100 protokolov). Architektúra vznikla postupne (v akademickom prostredí, neskôr sa rozšírila aj do komerčnej sféry) – najprv vznikli protokoly, potom vrstvy – a od vzniku sa toho zmenilo len málo (zmeny sú aditívne). Je to najpoužívanejšia sieťová technológia (IP over everything, everything over IP). Prístup autorov bol, na rozdiel od ISO/OSI, od jednoduchšieho k zložitejšiemu – najprv sa vytvárajú jednoduché riešenia, ktoré sa postupne obohacujú. Až sa riešenie prakticky overí (2 nezávislé implementácie), vznikne štandard. TCP/IP predpokladá že siete sú typu nespojované, nespoľahlivé a best effort. Všetka inteligencia je sústredená do koncových uzlov, sieť je „hlúpa“ ale rýchla.

TCP/IP bol pôvodne určený pre ARPAnet – nemohol mať teda žiadnu centrálnu časť a musel byť robustný voči chybám (nespoľahlivé/nespojované prenosy). Dôraz sa kládol aj na "internetworking". Nebolo však požadované zabezpečenie, mobilita ani kvalita služieb.

TCP/IP nedefinuje rôzne siete (čo sa hardvérových vlastností týka) a technológie vo vrstve sieťového rozhrania – iba sa snaží nad nimi prevádzkovať protokol IP

(okrem SLIP a PPP pre dvojbodové spoje). V sieťovej vrstve je IP protokol, v transportnej jednotné transportné protokoly (TCP a UDP), v aplikačnej potom jednotné základy aplikácií (email, prenos súborov, remote login...).

Adresace, IPv4, IPv6

Data se v IP síti posílají po blocích nazývaných datagramy. Jednotlivé datagramy putují sítí zcela nezávisle, na začátku komunikace není potřeba navazovat spojení či jinak „připravovat cestu“ datům, přestože spolu třeba příslušné stroje nikdy předtím nekomunikovaly.

IP protokol v doručování datagramů poskytuje nespolehlivou službu, označuje se také jako best effort – „nejlepší úsilí“; tj. všechny stroje na trase se datagram snaží podle svých možností poslat blíže k cíli, ale nezaručují prakticky nic. Datagram vůbec nemusí dorazit, může být naopak doručen několikrát a neručí se ani za pořadí doručených paketů. Pokud aplikace potřebuje spolehlivost, je potřeba ji implementovat v jiné vrstvě síťové architektury, typicky protokoly bezprostředně nad IP (viz TCP).

Pokud by síť často ztrácela pakety, měnila jejich pořadí nebo je poškozovala, výkon sítě pozorovaný uživatelem by byl malý. Na druhou stranu příležitostná chyba nemívá pozorovatelný efekt, navíc se obvykle používá vyšší vrstva, která ji automaticky opraví.

V **IPv4** je *adresou* 32bitové číslo, zapisované po jednotlivých bajtech, oddělených tečkami. Takových čísel existuje celkem 2^{32} . Určitá část adres je ovšem rezervována pro vnitřní potřeby protokolu a nemohou být přiděleny. Dále pak praktické důvody vedou k tomu, že adresy je nutno přidělovat hierarchicky, takže celý adresní prostor není možné využít beze zbytku. To vede k tomu, že v současnosti je již znatelný nedostatek IP adres, který řeší různými způsoby: dynamickým přidělováním (tzn. např. každý uživatel dial-up připojení dostane dočasnou IP adresu ve chvíli, kdy se připojí, ale jakmile se odpojí, je jeho IP adresa přidělena někomu jinému; při příštím připojení pak může tentýž uživatel dostat úplně jinou adresu), překladem adres (NAT) a podobně. Ke správě tohoto přidělování slouží specializované síťové protokoly, jako např. DHCP.

Pôvodný koncept adres počítal so štruktúrou adresy IPv4 v tvare *sieť:počítač*, kde bolo delenie častí pevne dané. Neskôr sa to ale ukázalo ako príliš hrubé delenie a lokálna časť adresy (v rámci jednej podsiete) môže mať dnes promennivú dĺžku. Obecne platí, že medzi adresami v rovnakej podsieti (majú rovnakú sieťovú časť) je možné dopravovať dáta priamo – dotyční účastníci sú prepojení jedným ethernetom alebo inou lokálnou sieťou. V opačnom prípade sa dáta dopravujú *smerničmi/routermi*. Hranicu v adrese medzi adresou siete a počítača určuje dnes maska podsiete. Jedná sa o 32 bitovú hodnotu, ktorá obsahuje jednotky tam, kde je v adrese určená sieť.

Adresovanie sietí bolo v prvopočiatkoch internetu vyriešené staticky – prvých 8 bitov adresy určovalo sieť, zvyšok jednotlivé počítače (existovať tak mohlo max. 256 sietí). S nástupom lokálnych sietí bolo tento systém potrebné zmeniť – zaviedli sa *triedy IP adres*. Existovalo 5 tried (A(začiatok 0, hodnoty prvého bajtu 0-127, maska 255.0.0.0), B(10, 128-191, 255.255.0.0), C(110, 192-223, 255.255.255.0), D(1110, 224-239, určené na multicast) a E(1111, 240-255, určené ako rezerva)). Po-

stupom času sa ale aj toto rozdelenie ukázalo ako nepružné a bol zavedený CIDR (Classless Inter-Domain Routing) systém v ktorom je možné hranicu medzi adresou siete a lokálnou časťou adresy umiestniť ľubovoľne (označuje sa potom ako kombinácia prefixu a dĺžky vo forme 192.168.0.0/24, kde 24 znamená že adresu tvorí prvých 24 bitov – jiný zápis je pomocí už zmiňované masky podsítě, tj. 192.168.0.0 s maskou 255.255.255.0).

Medzi adresami existujú niektoré tzv. **vyhradené adresy**, ktoré majú špeciálny význam.

- Adresa s (binárnymi) nulami v časti určujúcej počítač (192.168.0.0 (/24)) znamená „táto sieť“, resp. „táto stanica“.
- Adresa s jednotkami v časti určujúcej počítač (192.168.0.255 (/24)) znamená broadcast – všesmerové vysielanie.
- Adresy 10.0.0.0 – 10.255.255.255, 172.16.0.0 – 172.31.255.255 a 192.168.0.0 – 192.168.255.255 sa používajú na adresovanie interných sietí – smerovače tieto adresy nesmie smerovať ďalej do internetu.

IPv6 je trvalejším riešením nedostatku adres – zatiaľ sa ale rozširuje veľmi pozvoľna. Adresa v IPv6 má dĺžku 128 bitov (oproti 32), čo znamená cca. 6×10^{23} IP adres na $1m^2$ zemského povrchu – umožňuje teda, aby každé zariadenie na zemi malo vlastnú jednoznačnú adresu. Adresa IPv6 sa zapisuje ako osem skupín po štyroch hexadecimálnych číslach (napr. 2001:0718:1c01:0016:0214:22ff:fec9:0ca5) – pričom úvodné nuly v číslach je možné vynechať. Ak po sebe nasleduje niekoľko nulových skupín, je možné použiť len znaky :: – napr. ::1 miesto 0000:0000:.....:0001. Toto je možné použiť len raz v zápise adresy. RFC 4291 zavádza 3 typy adres:

- **individuálne / unicast** – identifikujú práve jedno rozhranie
- **skupinové / multicast** – určuje skupinu zariadení, ktorým sa má správa dopraviť
- **výberové / anycast** – určuje tiež skupinu zariadení, dáta sa však doručia len jednému z členov (najbližšiemu)

IPv6 neobsahuje všesmerové (broadcast) adresy. Byly nahrazeny obecnějším modelem skupinových adres a pro potřeby doručení dat všem zařízením připojeným k určité síti slouží speciální skupinové adresy (např. ff02::1 označuje všechny uzly na dané lince).

IPv6 zavádí také koncepci dosahu (scope) adres. Adresa je jednoznačná vždy jen v rámci svého dosahu. Nejčastější dosah je pochopitelně globální, kdy adresa je jednoznačná v celém Internetu. Kromě toho se často používá dosah linkový, definující jednoznačnou adresu v rámci jedné linky (lokální síť, např. Ethernetu). Propracovanou strukturu dosahů mají skupinové adresy (viz níže).

Adresní prostor je rozdělen následovně:

prefix	význam
::/128	neurčená
::1/128	smyčka (loopback)
ff00::/8	skupinové
fe80::/10	individuální lokální linkové
ostatní	individuální globální

Výběrové adresy nemají rezervovanou svou vlastní část adresního prostoru. Jsou promíchány s individuálními a je otázkou lokální konfigurace, aby uzel poznal, zda se jedná o individuální či výběrovou adresu.

Strukturu globálních individuálních IPv6 adres definuje RFC 3587. Je velmi jednoduchá a de facto odpovídá (až na rozměry jednotlivých částí) výše uvedené struktuře IPv4 adresy.

n bitů	64-n bitů	64 bitů
globální směrovací prefix	adresa podsítě	adresa rozhraní

Globální směrovací prefix je de facto totéž co adresa sítě, následuje adresa podsítě a počítače (přesněji síťového rozhraní). V praxi je adresa podsítě až na výjimky 16bitová a globální prefix 48bitový. Ten je pak přidělován obvyklou hierarchií, jejíž stávající pravidla jsou:

- první dva bajty obsahují hodnotu 2001 (psáno v šestnáctkové soustavě)
- další dva bajty přiděluje regionální registrátor (RIR)
- další dva bajty přiděluje lokální registrátor (LIR)

Reálná struktura globální individuální adresy tedy vypadá následovně:

16 bitů	16 bitů	16 bitů	16 bitů	64 bitů
2001	přiděluje RIR	přiděluje LIR	adresa podsítě	adresa rozhraní

Adresa rozhraní by pak měla obsahovat modifikovaný EUI-64 identifikátor. Ten získáte z MAC adresy jednoduchým postupem: invertuje se druhý bit MAC adresy a doprostřed se vloží dva bajty obsahující hodnotu fffe. Z ethernetové adresy 00:14:22:c9:0c:a5 tak vznikne identifikátor 0214:22ff:fec9:0ca5.

Adresy začínající hodnotou ff sú tzv. "skupinové adresy" – čtyři následující bity v nej obsahují příznaky, ďalšie štyri potom dosah (napr. interface-local, link-local, admin-local, site-local, organization-local, global...)

IPv6 ďalej podporuje QoS a bezpečnosť (IPsec).

Routing

Pojmem **směrování** (routing, routování) je označováno hledání cest v počítačových sítích. Jeho úkolem je dopravit datový paket určenému adresátovi, pokud možno co nejefektivnější cestou. Síťová infrastruktura mezi odesílatelem a adresátem paketu může být velmi složitá. Směrování se proto zpravidla nezabývá celou cestou paketu, ale řeší vždy jen jeden krok – komu data předat jako dalšímu (tzv. „distribuované směrování“). Ten pak rozhoduje, co s paketem udělat dál.

V prípade, že je cieľová stanica packetu v rovnakej sieti ako je odosielateľ, o doručenie sa postará linková vrstva. V opačnom prípade musí odosielateľ určiť najvhodnejší odchodzí smer a poslať datagram smerovaču vo zvolenom smere.

Základní datovou strukturou pro směrování je směrovací tabulka (routing table). Představuje vlastně onu sadu ukazatelů, podle kterých se rozhoduje, co udělat s kterým paketem. Směrovací tabulka je složena ze záznamů obsahujících:

- cílovou adresu, které se dotýčný záznam týká. Může se jednat o adresu individuálního počítače, častěji však je cíl definován prefixem, tedy začátkem adresy. Prefix mívá podobu 147.230.0.0/16. Hodnota před lomítkem je adresa cíle, hodnota za lomítkem pak určuje počet významných bitů adresy. Uvedenému prefixu tedy vyhovuje každá adresa, která má v počátečních 16 bitech (čili prvních dvou bajtech) hodnotu 147.230.
- akci určující, co provést s datagramy, jejichž adresa vyhovuje prefixu. Akce mohou být dvou typů: doručit přímo adresátovi (pokud je dotýčný stroj s adresátem přímo spojen) nebo předat některému ze sousedů (jestliže je adresát vzdálen).

Směrovací rozhodnutí pak probíhá samostatně pro každý procházející datagram. Vezme se jeho cílová adresa a porovná se směrovací tabulkou následovně:

- Z tabulky se vyberou všechny vyhovující záznamy (jejichž prefix vyhovuje cílové adrese datagramu).
- Z vybraných záznamů se použije ten s nejdelším prefixem. Toto pravidlo vyjadřuje přirozený princip, že konkrétnější záznamy (jejichž prefix je delší, tedy přesnější; speciálním případem je *host-specific route*) mají přednost před obecnějšími (co může být např. i *default route*; ps: *agregace*).

Zajímavou otázkou je, jak vznikne a jak je udržována směrovací tabulka. Tento proces mají obecně na starosti směrovací algoritmy. Když jsou pak pro určitý algoritmus definována přesná pravidla komunikace a formáty zpráv nesoucích směrovací informace, vznikne směrovací protokol (routing protocol). Směrovací algoritmy můžeme rozdělit do dvou základních skupin: na statické a dynamické. Často se také mluví o statickém a dynamickém směrování, které je důsledkem činnosti příslušných protokolů.

Při **statickém (též neadaptivním) směrování** se směrovací tabulka nijak nemění. Je dána konfigurací počítače a případné změny je třeba v ní provést ručně. Tato varianta vypadá jako nepříliš atraktivní, ve skutečnosti ale drtivá většina zařízení v Internetu směřuje staticky.

Dynamické (adaptivní) směrování průběžně reaguje na změny v síťové topologii a přizpůsobuje jim směrovací tabulky. Na vytváření tabulek existuje několik algoritmů – routovacích protokolů (vector-distance/link-state) – RIP, BGP, OSPF.

Distribuované směrování

V distribuovaném směrování může výpočet cesty (směru předání paketu) provádět buď každý uzel nezávisle, nebo mohou uzly kooperovat (distribuovaný výpočet). Rozlišuje se také četnost aktualizace informací. Dva základní algoritmy distribuovaného směrování jsou:

- *vector distance* – každý uzel si udržuje tabulku vzdáleností, přímí sousedé si vyměňují informace o cestách ke všem uzlům, tj. jde o distribuovaný výpočet, přenáší se dost informací. Trpí problémem „count-to-infinity“ – tj. když 1 uzel přestane existovat, postupně si jeho sousedé mezi sebou přehazují vzdálenost, postupně o 1 zvětšovanou (do nekonečna). Řeší se pomocí technik „split

horizon“ (neinzeruj vzdálenost zpět) a „poisoned reverse“ (inzeruj zpět nekonečno), někde ale přesto selhává.

- *link state* – každý uzel hledá změny svých sousedů a pokud k nějaké dojde, pošle floodem informaci do celé sítě. Výpočet vzdáleností dělá každý uzel sám.

Tyto algoritmy se používají u některých známých směrovacích protokolů:

- *RIP* (Routing Information Protocol) – protokol z BSD Unixu, typu vector distance. Počítá s max. 16 přeskoky, změny se updatují 2x za minutu. Informace ve směrovací tabulce může zahrnovat max. 25 sítí, používá split horizon & poisoned reverse. Hodí se ale jen pro malé sítě.
- *OSPF* (Open Shortest Path First) – jde o protokol typu link state, uzly si počítají vzdálenosti do všech sítí Dijkstrovým algoritmem. Pro zjišťování změn se posílají pakety „HELLO“ a „ECHO“. Má lepší škálovatelnost, hodí se pro větší sítě.

Hierarchické směrování, autonomní systémy

Hierarchické směrování znamená rozdělení sítě do oblastí (*areas*) a směrování mezi nimi jen přes vstupní body. Je vhodné pro velké, složitě propojené nebo různým způsobem spravované sítě. Nad oblastmi se vytvoří propojení – *backbone area* (páteřní systém), přes které se směrování mezi oblastmi provádí. Celému tomuto (areas + backbone area) se říká *autonomní systém*. Detailní směrovací informace neopouštějí jednotlivé oblasti.

Pro směrování v rámci jedné oblasti i mezi oblastmi v rámci jednoho autonomního systému slouží jeden z tzv. *interior gateway protocols*, může být použit např. OSPF nebo RIP, případně další jako IGRP (interior gateway routing protocol, typu vector distance) nebo EIGRP (enhanced IGRP, hybrid mezi vector distance a link state). Mezi jednotlivými autonomními systémy (přes AS boundary routers) se směruje pomocí *exterior gateway protocolu*, jedním z nich je např. *Border Gateway Protocol* (BGP).

Díky existenci autonomních systémů jde např. při peeringu stanovit, který provoz půjde přes peering a který výše po upstreamu do páteřních sítí.

Fragmentace

Maximum transmission unit (MTU) je maximální velikost paketu, který je možné přenést z jednoho síťového zařízení na druhé. Obvyklá hodnota MTU v případě Ethernetu je cca 1500 bajtů, nicméně mezi některými místy počítačové sítě (spojených například modemem nebo sériovou linkou) může být maximální délka přeneseného paketu nižší. Hodnotu MTU lze zjistit prostřednictvím protokolu ICMP. Při posílání paketů přes několik síťových zařízení je samozřejmě důležité nalézt nejmenší MTU na dané cestě. Hodnota MTU je omezena zdola na 576 bajtů.

U přenosového protokolu TCP je při směrování paketu do přenosového kanálu s nižším MTU než je délka paketu, provedena **fragmentace paketu**. U protokolu UDP není fragmentace paketu podporována a paket je v takovém případě zahozen.

Pokud dorazí na směrovač paket o velikosti větší, než kterou je přenosová trasa schopna přenést (např. při přechodu z Token Ringu používajícího 4 kByte pakety

na Ethernet používajícího maximálně 1,5 kByte pakety), musí směrovač zajistit tzv. fragmentaci, neboli rozebrání paketu na menší části a cílový uzel musí zajistit opětovné složení, neboli defragmentaci.

Fragmenty procházejí přes síť jako samostatné datagramy. Aby byl koncový uzel schopen fragmenty složit do originálního datagramu, musí být fragmenty příslušně označeny. Toto označování se provádí v příslušných polích IP hlavičky.

Pokud nesmí být datagram fragmentován, je označen v příslušném místě IP hlavičky příznakem „Don't Fragment“. Jestliže takto označený paket dorazí na směrovač, který by jej měl poslat prostředím s nižším MTU a tudíž je nutnost provést fragmentaci, provede směrovač jeho zrušení a informuje odesílatele chybovou zprávou ICMP.

Aby byl cílový uzel schopen složit originální datagram, musí mít dostatečný buffer do něhož jsou jednotlivé fragmenty ukládány na příslušnou pozici danou offsetem. Složení je dokončeno v okamžiku, kdy je vyplněn celý datagram začínající fragmentem s nulovým offsetem (identification a fragmentation offset v hlavičce) a končící segmentem s příznakem „More Data Flag“ (resp. More Fragments) nastaveným na False.

V IPv4 je možné fragmentované pakety dálej deliť; naproti tomu v IPv6 musí fragmentáciu zabezpečiť odosielateľ – nevyhovujúce pakety sa zahadzujú.

Spolehlivost, Flow control, Congestion control

Keďže TCP/IP funguje nad obecně nespojovanými a nespoľahlivými médiami, **spoľahlivosť** ktorú TCP poskytuje nie je „skutočná“, ale len „softvérovo emulovaná“ – medziľahlé uzly o spojení nič nevedia, fungujú nespojovane (pre komunikáciu sa používa sieťová vrstva, transportná „existuje“ iba medzi koncovými uzlami). Je teda nutné ošetriť napr. nespoľahlivosť infraštruktúry (strácanie dát, duplicity – pričom stratiť sa môže aj žiadosť o vytvorenie pripojenia, potvrdenie...) a reboot uzlov (uzol stratí históriu, je potrebné ošetriť existujúce spojenia...).

Používa sa celá rada techník, kde základom je kontinuálne potvrdzovanie: príjemca posiela kladné potvrdenia; odosielateľ po každom odoslaní spúšťa časovač a ak mu do vypršania nepríde potvrdenie, posiela dáta znovu. Potvrdzovanie nie je samostatné ale vkladá sa do paketov cestujúcich opačným smerom – *piggybacking*.

TCP priebežne kontroluje „dobu obrátky“ a vyhodnocuje vážený priemer a rozptyl dôb obrátky. Čakaciu dobu (na potvrdenie) potom vypočítava ako funkciu tohto váženého priemeru a rozptylu. Výsledný efekt je potom ten, že čakacia doba je tesne nad strednou dobou obrátky. V prípade konštantnej doby obrátky sa čakacia doba približuje strednej dobe obrátky; ak kolíše, čakacia doba sa zväčšuje.

Dáta v TCP sa prijímajú/posielajú po jednotlivých byteoch – interne sa však bufferujú a posielajú až po naplnení buffera (pričom aplikácia si môže vyžiadať okamžité odoslanie – operácia PUSH). TCP si potrebuje označovať jednotlivé byty v rámci prúdu (keďže nepracuje s blokmi) – napr. kvôli potvrdzovaniu; používa sa na to 32-bitová pozícia v bytovom prúde (začína sa od náhodne zvoleného čísla).

TCP sa snaží **riadiť tok dát** – aby odosielateľ nezahľcoval príjemcu a kvôli tomu nedochádzalo k strate dát. Podstata riešenia je tzv. *metóda okienka*. Okienko udáva veľkosť voľných bufferov na strane prijímajúceho a odosielateľ môže posielateľ dáta až do „zaplnenia“ okienka. Príjemca spolu s každým potvrdením posiela aj

svoju ponuku – údaj o veľkosti okienka (window advertiment)., ktorý hovorí koľko ešte dát je schopný prijať (naviac k práve potvrdeným). Znovu – používa sa metóda kontinuálneho potvrdzovania.

Väčšina strát prenášaných dát ide skôr na vrub zahlteniu ako chybám HW a transportné protokoly môžu nevhodným chovaním zhoršovať dôsledky. TCP každú stratu dát chápe ako dôsledok zahltenia – nasadzuje **opatrenia proti zahlteniu** (congestion control). Po strate paketu ho pošle znovu ale neposiela ďalšie a čaká na potvrdenie (tj. prechod z kontinuálneho potvrdzovania na jednotlivé \Rightarrow vysiela menej dát ako mu umožňuje okienko). Ak príde potvrdenie včas, zdvojnásobí množstvo odosielaných dát – a tak pokračuje kým nenarazí na aktuálnu veľkosti okienka (postupne sa tak vracia na kontinuálne potvrdzovanie).

Dôležitou vlastnosťou je aj korektné chovanie pri navázovaní a rušení spojenia (v prostredí, kde môže dôjsť k spomaleniu, strate, duplicite...) – používa sa tzv. 3-fázový handshake. Vytvorenie spojenia prebieha nasledovne:

1. Klient pošle serveru SYN paket (v pakete je nastavený príznak SYN) spolu s náhodným *sequence number* (X).
2. Server tento paket prijme, zaznamená si *sequence number* (X) a pošle späť paket SYN-ACK. Tento paket obsahuje pole Acknowledgement, ktoré označuje ďalšie číslo (*sequence number*), ktoré tento host očakáva (X+1). Tento host rovno vytvorí spätnú session s vlastným sekvenčným číslom (Y).
3. Klient odpovie so sekvenčným číslom (X+1) a jednoduchým Acknowledgement číslom (Y+1) – čo je sekvenčné číslo servera+1.

Pak už spojenie považované za navázané. Rušenie spojenia funguje podobne, posílajú sa pakety FIN (finish), FIN+ACK a ACK. Pokud více než nějaký určitý počet pokusů o odeslání (po spočítaných time-outech) jednoho z 3-way handshake paketů selže (druhá strana neodešle to, co mělo následovat), spojení se považuje za přerušené (i u navazování, i u rušení).

NAT

TODO: přeložit ty copy & paste z Wiki

Network address translation (zkráceně NAT, česky překlad síťových adres) je funkce síťového routeru pro změnu IP adres paketů procházejících zařízením, kdy se zdrojová nebo cílová IP adresa převádí mezi různými rozsahy. Nejběžnější formou je tzv. maškaráda (maskování), kdy router IP adresy z nějakého rozsahu mění na svoji IP adresu a naopak – tím umožňuje, aby počítače ve vnitřní síti (LAN) vystupovaly v Internetu pod jedinou IP adresou. Router si drží po celou dobu spojení v paměti tabulku překladu adres.

Překlad síťových adres je funkce, která umožňuje překládání adres. Což znamená, že adresy z lokální sítě přeloží na jedinečnou adresu, která slouží pro vstup do jiné sítě (např. Internetu), adresu překládanou si uloží do tabulky pod náhodným portem, při odpovědi si v tabulce vyhledá port a pošle pakety na IP adresu přiřazenou k danému portu. NAT je vlastně jednoduchým proxy serverem (na síťové vrstve).

Komunikace

Klient odešle požadavek na komunikace, směrovač se podívá do tabulky a zjistí, zdali se jedná o adresu lokální, nebo adresu venkovní. V případě venkovní adresy si do tabulky uloží číslo náhodného portu, pod kterým bude vysílat a k němu si přiřadí IP adresu. Během přeposílání „ven“ a změny adresy v paketu musí NAT také přepočítat CRC checksum TCP i IP (aby pakety nebyly zahazovány kvůli špatnému CRC, protože změněná adresa je jejich součástí).

Výhodami NAT sú umožnenie pripojenie viacerých počítačov do internetu cez jednu zdieľanú verejnú IP adresu, a zvýšenie bezpečnosti počítačov za NATom (aj keď je to security through obscurity a nie je dobré postaviť bezpečnosť iba na NATe). Nevýhodami potom sú nefungujúce protokoly (napr. aktívne FTP) – čo je zrejmé z fungovania NATu.

NAT Traversal

NAT traversal refers to an algorithm for the common problem in TCP/IP networking of establishing connections between hosts in private TCP/IP networks that use NAT devices.

This problem is typically faced by developers of client-to-client networking applications, especially in peer-to-peer and VoIP activities. NAT-T is commonly used by IPsec VPN clients in order to have ESP packets go through NAT.

Many techniques exist, but no technique works in every situation since NAT behavior is not standardized. Many techniques require a public server on a well-known globally-reachable IP address. Some methods use the server only when establishing the connection (such as STUN), while others are based on relaying all the data through it (such as TURN), which adds bandwidth costs and increases latency, detrimental to conversational VoIP applications.

Druhy uspořádání NATu

- *Static NAT*: A type of NAT in which a private IP address is mapped to a public IP address, where the public address is always the same IP address (i.e., it has a static address). This allows an internal host, such as a Web server, to have an unregistered (private) IP address and still be reachable over the Internet.
- *Dynamic NAT*— A type of NAT in which a private IP address is mapped to a public IP address drawing from a pool of registered (public) IP addresses. Typically, the NAT router in a network will keep a table of registered IP addresses, and when a private IP address requests access to the Internet, the router chooses an IP address from the table that is not at the time being used by another private IP address. Dynamic NAT helps to secure a network as it masks the internal configuration of a private network and makes it difficult for someone outside the network to monitor individual usage patterns. Another advantage of dynamic NAT is that it allows a private network to use private IP addresses that are invalid on the Internet but useful as internal addresses.
- *PAT* — PAT (NAT overloading) je další variantou NATu. U této varianty NATu se více inside local adres mapuje na jednu inside global adresu na různých portech. Tedy máme jednu veřejnou adresu a vnitřní síť oadresovanou

inside local adresami. Překladová tabulka je rozšířena o dvě položky: inside local port – port, ze kterého byl paket odeslán a inside global port – číslo portu, na který je paket odesláný ze zdrojového portu počítače mapován. Výhodou je, že se tak připojuje více počítačů přes jednu IP adresu.

ARP

Address Resolution Protocol (ARP) se v počítačových sítích s IP protokolem používá k získání ethernetové (MAC) adresy sousedního stroje z jeho IP adresy. Používá se v situaci, kdy je třeba odeslat IP datagram na adresu ležící ve stejné podsíti jako odesílatel. Data se tedy mají poslat přímo adresátovi, u něhož však odesílatel zná pouze IP adresu. Pro odeslání prostřednictvím např. Ethernetu ale potřebuje znát cílovou ethernetovou adresu.

Proto vysílající odešle ARP dotaz (ARP request) obsahující hledanou IP adresu a údaje o sobě (vlastní IP adresu a MAC adresu). Tento dotaz se posílá linkovým broadcastem – na MAC adresu identifikující všechny účastníky dané lokální sítě (v případě Ethernetu na ff:ff:ff:ff:ff:ff). ARP dotaz nepřekročí hranice dané podsítě, ale všechna k ní připojená zařízení dotaz obdrží a jako optimalizační krok si zapíše údaje o jeho odesílateli (IP adresu a odpovídající MAC adresu) do své ARP cache. Vlastník hledané IP adresy pak odešle tazateli ARP odpověď (ARP reply) obsahující vlastní IP adresu a MAC adresu. Tu si tazatel zapíše do ARP cache a může odeslat datagram.

Informace o MAC adresách odpovídajících jednotlivým IP adresám se ukládají do ARP cache, kde jsou uloženy do vypršení své platnosti. Není tedy třeba hledat MAC adresu před odesláním každého datagramu – jednou získaná informace se využívá opakovaně. V řadě operačních systémů (Linux, Windows XP) lze obsah ARP cache zobrazit a ovlivňovat příkazem arp.

Alternativou pro počítač bez ARP protokolu je používat tabulku přiřazení MAC adres IP adresám definovanou jiným způsobem, například pevně konfigurovanou. Tento přístup se používá především v prostředí se zvýšenými nároky na bezpečnost, protože v ARP se dá podvádět – místo skutečného vlastníka hledané IP adresy může odpovědět někdo jiný a stáhnout tak k sobě jeho data.

ARP je definováno v RFC 826. Používá se pouze pro IPv4. Novější verze IP protokolu (IPv6) používá podobný mechanismus nazvaný Neighbor Discovery Protocol (NDP, „objevování sousedů“).

Ačkoliv se ARP v praxi používá téměř výhradně pro překlad IP adres na MAC adresy, nebyl původně vytvořen pouze pro IP síť. ARP se může použít pro překlad MAC adres mnoha různých protokolů na síťové vrstvě. ARP byl také uzpůsoben tak, aby vyhodnocoval jiné typy adres fyzické vrstvy: například ATMARP se používá k vyhodnocení ATM NSAP adres v protokolu Classical IP over ATM.

ICMP

ICMP protokol (anglicky Internet Control Message Protocol) je jeden z jádrových protokolů ze sady protokolů internetu. Používají ho operační systémy po-

čítačů v síti pro odesílání chybových zpráv – například pro oznámení, že požadovaná služba není dostupná nebo že potřebný počítač nebo router není dosažitelný.

ICMP se svým účelem liší od TCP a UDP protokolů tím, že se obvykle nepoužívá síťovými aplikacemi přímo. Jedinou výjimkou je nástroj ping, který posílá ICMP zprávy „Echo Request“ (a očekává příjem zprávy „Echo Response“) aby určil, zda je cílový počítač dosažitelný a jak dlouho paketům trvá, než se dostanou k cíli a zpět.

ICMP protokol je součástí sady protokolů internetu definovaná v RFC 792. ICMP zprávy se typicky generují při chybách v IP datagramech (specifikováno v RFC 1122) nebo pro diagnostické nebo routovací účely. Verze ICMP pro IPv4 je známá jako ICMPv4. IPv6 používá obdobný protokol: ICMPv6.

ICMP zprávy se konstruují nad IP vrstvou; obvykle z IP datagramu, který ICMP reakci vyvolal. IP vrstva patřičnou ICMP zprávu zapouzdří novou IP hlavičkou (aby se ICMP zpráva dostala zpět k původnímu odesílateli) a obvyklým způsobem vzniklý datagram odešle. Například každý stroj (jako třeba mezilehlé routery), který forwarduje IP datagram, musí v IP hlavičce dekrementovat políčko TTL („time to live“, „zbývající doba života“) o jedničku. Jestliže TTL klesne na 0 (a datagram není určen stroji provádějícímu dekrementaci), router přijatý paket zahodí a původnímu odesílateli datagramu pošle ICMP zprávu „Time to live exceeded in transit“ („během přenosu vypršela doba života“).

Každá ICMP zpráva je zapouzdřená přímo v jediném IP datagramu, a tak (jako u UDP) ICMP nezaručuje doručení. Ačkoli ICMP zprávy jsou obsaženy ve standardních IP datagramech, ICMP zprávy se zpracovávají odlišně od normálního zpracování prokolů nad IP. V mnoha případech je nutné prozkoumat obsah ICMP zprávy a doručit patřičnou chybovou zprávu aplikaci, která vyslala původní IP paket, který způsobil odeslání ICMP zprávy k původci.

Mnoho běžně používaných síťových diagnostických utilit je založeno na ICMP zprávách. Příkaz traceroute je implementován odesláním UDP datagramů se speciálně nastavenou životností v TTL políčku IP hlavičky a očekáváním ICMP odezvy „Time to live exceeded in transit“ nebo „Destination unreachable“. Příbuzná utilita ping je implementována použitím ICMP zpráv „Echo“ a „Echo reply“.

Nejpoužívanější ICMP datagramy:

- *Echo*: požadavek na odpověď, každý prvek v síti pracující na IP vrstvě by na tuto výzvu měl reagovat. Často to z různých důvodů není dodržováno.
- *Echo Reply*: odpověď na požadavek
- *Destination Unreachable*: informace o nedostupnosti cíle, obsahuje další upřesňující informaci
 - Net Unreachable: nedostupná cílová síť, reakce směrovače na požadavek komunikovat se sítí, do které nezná cestu
 - Host Unreachable: nedostupný cílový stroj
 - Protocol Unreachable: informace o nemožnosti použít vybraný protokol
 - Port Unreachable: informace o nemožnosti připojit se na vybraný port
- *Redirect*: přesměrování, používá se především pokud ze sítě vede k cíli lepší cesta než přes defaultní bránu. Stanice většinou nepoužívají směrovací protokoly a proto jsou informovány touto cestou. Funguje tak, že stanice pošle

datagram své, většinou defaultní, bráně, ta jej přeošle správným směrem a zároveň informuje stanici o lepší cestě.

- Redirect Datagram for the Network: informuje o přesměrování datagramů do celé sítě
- Redirect Datagram for the Host: informuje o přesměrování datagramů pro jediný stroj
- *Time Exceeded*: vypršel časový limit
 - Time to Live exceeded in Transit: během přenosu došlo ke snížení TTL na 0 aniž byl datagram doručen
 - Fragment Reassembly Time Exceeded: nepodařilo se sestavit jednotlivé fragmenty v časovém limitu (např pokud dojde ke ztrátě části datagramů)

Ostatní datagramy jsou používány spíše vzácně, někdy je používání ICMP znemožněno zcela špatným nastavením firewallu.

UDP, TCP

UDP – nespolehlivý nespojovaný přenos datagramov... přidává len porty

TCP – porty+spolehlivý spojovaný přenos streamov...

...dalšie info viď kapitolu o BSD Sockets :-)

6.3 Rozhraní BSD Sockets

Úvod

Berkeley (BSD) sockets je rozhranie (API) na vyvíjanie aplikácií ktoré používajú medziprocesovú komunikáciu (napr. v rámci siete). De facto je to štandardná abstrakcia pre sieťové sockety. Primárnym jazykom tohto API je C, pre väčšinu ostatných však existujú podobné rozhrania.

BSD sockets je API umožňujúce komunikáciu medzi dvomi hostmi alebo procesmi na jednom počítači, používajúc koncepciu internetových socketov. Toto rozhranie je implicitné pre TCP/IP a je teda jednou zo základných technológií internetu. Programátori môžu využívať rozhrania socketov na troch úrovniach, najzákladnejšou z nich sú RAW sockety (aj keď túto úroveň sa využijú zväčša len na počítačoch implementujúcich technológie týkajúce sa už priamo internetu).

Hlavičkové súbory

Berkeley sockets používajú viaceré hlavičkové súbory, okrem iného:

- **sys/socket.h** Core BSD socket functions and data structures.
- **netinet/in.h** AF_INET and AF_INET6 address families. Widely used on the Internet, these include IP addresses and TCP and UDP port numbers.
- **sys/un.h** AF_UNIX address family. Used for local communication between programs running on the same computer. Not used on networks.
- **arpa/inet.h** Functions for manipulating numeric IP addresses.

- **netdb.h** Functions for translating protocol names and host names into numeric addresses. Searches local data as well as DNS.

TCP

TCP poskytuje koncept spojenia. Proces vytvorí TCP socket pomocou volania `socket()` s parametrom `PF_INET(6)` a `SOCK_STREAM`.

Server

Vytvorenie jednoduchého TCP servera vyžaduje nasledujúce kroky:

- Vytvorenie TCP socketu (pomocou volania `socket()`)
- Pripojenie socketu na port, kde bude načúvať (`bind()`); parametrami je `sockaddr_in` štruktúra, v ktorej sa nastavuje `sin_family` (`AF_INET-IPv4`, `AF_INET6-IPv6`) a `sin_port`)
- Pripravenie socketu na načúvanie na porte (`listen()`).
- Akceptovanie prichádzajúcich pripojení pomocou `accept()`. Táto funkcia blokuje volajúceho do príchodu pripojenia a vracia identifikátor prichádzajúceho spojenia, ktorý sa môže ďalej použiť. `accept()` je hneď možné volať na pôvodný identifikátor socketu na čakanie na ďalšie spojenia.
- Komunikácia s klientom pomocou `send()`, `recv()` alebo `read()` a `write()`
- Keď už socket nie je potrebný, je možné ho zavrieť pomocou `close()`.

Klient

Vytvorenie TCP klienta vyžaduje nasledujúce kroky:

- Vytvorenie TCP socketu (pomocou volania `socket()`)
- Pripojenie k serveru pomocou `connect()` (znovu sa používa štruktúra `sockaddr_in`, vyplní sa `sin_family`, `sin_port` (ako pri serveri) + `sin_addr` (adresa servera))
- Komunikácia so serverom pomocou `send()`, `recv()` alebo `read()` a `write()`
- Keď už socket nie je potrebný, je možné ho zavrieť pomocou `close()`.

UDP

UDP je protokol bez spojenia (connectionless) a bez garancie doručenia správ. UDP balíky môžu (okrem správneho počtu/poradia) doraziť mimo poradia, môžu byť duplikované alebo nedoraziť ani raz. Vďaka minimálnym garanciám má UDP oproti TCP oveľa menšiu réžiu. Keďže tento protokol nevytvára spojenia, dáta sa prenášajú v datagramoch.

Adresovací priestor UDP (porty UDP) je úplne nezávislý na priestore portov TCP.

Server

Keďže sa nevytvárajú spojenia, po vytvorení socketu (ako pri TCP pomocou `socket()+bind()`) už aplikácia (server) rovno čaká prichádzajúce datagramy pomocou funkcie `recvfrom()`. Na konci sa socket zatvára pomocou `close()`.

Klient

U klienta je tiež oproti spojovanej verzii zjednodušenie - stačí vyrobiť socket (pomocou `socket()`) a potom už iba posielat datagramy pomocou `sendto()`. Na konci sa socket zatvára pomocou `close()`.

Najdôležitejšie funkcie

- **int socket(int domain, int type, int protocol)**
 - *domain* (PF_INET — PF_INET6)
 - *type* (SOCK_STREAM, SOCK_DGRAM, SOCK_SEQPACKET (spoľahlivé zoradené balíky), SOCK_RAW (raw protokoly nad sieťovou vrstvou))
 - *protocol* (väčšinou IPPROTO_IP, ďalšie sú v `netinet/in.h`)
- **struct hostent *gethostbyname(const char *name)**
struct hostent *gethostbyaddr(const void *addr, int len, int type)
 - Vracia pointer na `hostent` štruktúru, ktorá popisuje internetového hosta zadaného pomocou mena alebo adresy (obsahuje buď informácie od name servera, alebo z lokálneho `/etc/hosts` súboru)...
- **int connect(int sockfd, const struct sockaddr *serv_addr, socklen_t addrlen)**
- **int bind(int sockfd, struct sockaddr *my_addr, socklen_t addrlen)**
- **int listen(int sockfd, int backlog)**
 - *backlog* určuje maximálne koľko pripojení môže vo fronte čakať na akceptovanie...
- **int accept(int sockfd, struct sockaddr *cliaddr, socklen_t *addrlen)**
do *cliaddr* sa vyplnia informácie o klientovi...

Blokujúce a neblokujúce volania

BSD sockety môžu fungovať v dvoch módoch - blokujúcich a neblokujúcich. V blokujúcom móde funkcie nevrátia riadenie programu, kým nie sú spracované všetky dáta - čo môže spôsobiť rôzne problémy (program „zamrzne“, keď socket načúva; alebo keď socket čaká na dáta, ktoré neprichádzajú). Typicky sa nastavuje neblokujúci mód pomocou `fcntl()` alebo `ioctl()`

6.4 Spolehlivosť - spojované a nespojované protokoly, typy, detekcie a oprava chýb

Spolehlivosť

Spolehlivosť:

- môže byť zajištená na ktorékoliv vrstve (kromě fyzické)
- TCP/IP řeší na transportní (TCP), ISO/OSI očekává spolehlivost na všech (počínaje linkovou)

- větší režie, zpoždění při chybách

Nespolehlivá komunikace:

- menší režie, lepší odezva
- výhodné pro audio/video přenosy, kde lze tolerovat ztráty

Spojované a nespojované protokoly

Spojovaná komunikace: stavová, virtuální okruhy, navazování a ukončení spojení. Viz TCP.

Nespojovaná komunikace: zasílání zpráv, datagramy (UDP), nestavová, bez navazování a ukončování. Viz UDP.

Detekce a oprava chyb

- schopnost poznat, že došlo k nějaké chybě při přenosu
- Hammingovy kódy - příliš velká redundance, nepoužívané
- potvrzování (ACK) - viz TCP/IP
 - příjemce si znovu nechá zaslat poškozená/nedoručená data
 - podmínkou existence zpětného kanálu (alespoň half-duplex)
 - jednotlivé vs. kontinuální
 - kladné (ACK) a záporné (NAK)
 - samostatné vs. nesamostatné (piggybacking)
 - metoda okénka
 - selektivní opakování vs. opakování s návratem
- parita - příčná, podélná
- kontrolní součty
- cyklické redundantní součty (CRC)
- druhy chyb: pozměněná data, shluky chyb, výpadky dat
- při chybě nutno vyžádat si celý rámeček znovu

6.5 Bezpečnost – IPSec, principy fungování AH, ESP, transport mode, tunnel mode, firewalls

IPSec

- Není to pouze jeden protokol ale soustava vzájemně provázaných opatření a dílčích protokolů pro zabezpečení komunikace pomocí IP protokolu, funguje na síťové vrstvě – není závislý na protokolech vyšších vrstev jako je TCP a UDP (např. SSL protokol pracuje na transportní vrstvě)
- Podporováno jak v IPv4 (podpora nepovinná) i v IPv6 (podpora povinná)
- Zajišťuje důvěrnost (šifruje přenášená data) a integritu (data nejsou při přenosu změněna)
- několik desítek RFC dokumentů
- autentifikace – ověření původu dat (odesílatele)

- kryptování – šifrování komunikace (mimo IP hlavičky)
- může být implementováno na bráně (security gateway, lokální síť je považována za bezpečnou) nebo na koncových zařízeních
- **SA (Security Association)**
 - point-to-point bezpečnostní spoj (návrh uvažuje i o jiných variantách)
 - pro každý směr a každý protokol nutné mít vlastní SA spoj

IPsec módy:

- **transport mode**
 - IP hlavička nechráněná (jeden z důvodů je užívání systému NAT), tělo paketu šifrováno (data vyšších protokolů)
 - použitelné jen na koncových stanicích
- **tunnel mode**
 - pakety jsou celé (včetně hlavičky) zašifrovány a vloženy do dalšího paketu, na druhé straně rozbaleny
 - povinné pro security gateways, volitelné pro koncové stanice
 - ve vnější IP hlavičce se jako příjemce uvádí security gateway na hranici cílové sítě

IPsec protokoly:

- **AH (Authentication Header)**
 - komunikující strany se dohodnou na klíči
 - k datům se připojuje hash
 - chrání také před replay attack
 - provádí autentizaci a kontrolu změny dat, neprovádí šifrování
- **ESP (Encapsulating Security Payload)**
 - provádí autentizaci a také šifruje obsah
 - pro šifrování používá 3DES, Blowfish aj. (původně DES, již není považováno za bezpečné)

Dohoda klíčů:

- před použitím protokolu AH či ESP si musí strany dohodnout klíče
- manuální konfigurace
- automatická konfigurace – IKE (Internet Key Exchange) protokol

Firewally

- sledování a filtrování komunikace na síti
 - blokování – zabraňuje neoprávněnému přístupu
 - prostupnost – propouštění povoleného toku
- paketové filtry – např. na routeru
- stavový firewall (stateful) – sleduje vztahy mezi pakety, ohlíží se na historii

- na různých vrstvách
 - síťová – pouze dle zdrojových a cílových adres a protokolu
 - transportní – také podle portů
 - aplikační – dle obsahu (dat)
- demilitarizovaná zóna (DMZ):
 - jiné řešení bezpečnosti
 - přístup ven pouze přes specializovaná zařízení (proxy, brány), nelze přímo
 - platí pro oba směry

6.6 Internetové a intranetové protokoly a technologie – DNS, SMTP, FTP, HTTP, NFS, HTML, XML, XSLT a jejich použití

DNS (Domain Name System)

- Řešení které umožňuje používat symbolická jména místo číselných adres realizované pomocí DNS serverů, distribuované řešení – výpadek i více serverů nevyřadí službu z provozu
- hierarchický (stromový) prostor jmen. Kořenem je tzv. kořenová doména — tečka, pod ní se v hierarchii nachází domény nejvyšší úrovně (com, edu, cz, uk, ...), strom je možné rozdělit do zón a její správu svěřit někomu dalšímu – právě možnost delegování pravomocí a distribuovaná správa tvoří klíč. vlastnosti DNS a stojí za jeho úspěchem
- DNS servery
 - Primární – zde data vznikají, zde se musí také provádět změny, každá doména obsahuje právě jeden
 - Sekundární -- automatická kopie primárního, průběžně si aktualizuje data, slouží jednak jako záloha pro případ výpadku prim. serveru a také pro rozkládání zátěže, každá doména musí mít alespoň jeden sekundární server
 - Pomocný (caching only) -- slouží jako vyrovnávací paměť pro snížení zátěže celého systému, uchovává si odpovědi a poskytuje je při opakování dotazů dokud jim nevyprší životnost
- Odpovědi z primárního a ze sekundárních serverů jsou autoritativní – platné. Z pomocného serveru je odpověď neautoritativní – klient může požádat o autoritativní odpověď.
- soubor hosts
- doména, zóna, delegace, TLD, ccTLD, gTLD
- syntaxe jmen, FQDN (plně kvalifikované...)
- IDN
- iterativní dotaz, rekurzivní dotaz, cachování, resolvery, TTL, autoritativní odpověď
- Resource Records (RR) – jednotka informace v DNS
 - formát: [name type class TLL rdlength rdata]

- class – dnes vždy IN (internet)
- TTL – time to live (doba platnosti záznamu)
- types – A (IPv4 adresa), NS (hostname nameserveru), MX (mailserver), PTR (domain name pointer – reverse DNS), AAAA (IPv6 adresa), SPF, TXT, SRV, ...
- rlength – délka dat, rdata – vlastní data
- DNS protokol – TCP/UDP, truncation

SMTP (Simple Mail Transfer Protocol)

- Internetový protokol určený pro přenos zpráv elektronické pošty mezi stanicemi. Protokol zajišťuje doručení pošty pomocí přímého spojení mezi odesílatelem a adresátem; zpráva je doručena do tzv. poštovní schránky adresáta, ke které potom může uživatel kdykoli (offline) přistupovat a vybírat zpráva pomocí protokolů POP3 popř. IMAP.
- Funguje nad protokolem TCP, používá port 25
- Pro netextové přenosy je využíván standart MIME (řeší problém národních abeced, formátování, příloh ...) – rozšíření formátu zpráv – Quoted-Printable, Base64, mime-type
- SMTP – protokol pro přenos zpráv
- RFC822 – formát zpráv – 7-bitová data
- POP3, IMAP – stahování zpráv ze schránky
- struktura zprávy – hlavička + tělo + přílohy
- e-mailové adresy, MX záznamy, priority

FTP (File Transfer Protocol)

- Jeden ze sady protokolů TCP/IP, netransparentní řešení – uživatel si uvědomuje že se soubor nachází na vzdáleném počítači, typicky se vzdálený soubor celý přenesou na místní počítač a zde se s ním pracuje ; v rámci TCP/IP existuje zjednodušená verze – TFTP
- Dva režimy – textový a binární ; vychází z modelu klient/server, klient je typicky aplikační program, server je obvykle systémový proces (démon,...)
- Jednotný formát pro potřeby přenosu dat, veškeré konverze provádí koncové uzly
- Používají se 2 různá spojení – řídicí (přenos příkazů – FTP má vlastní řídicí jazyk) a datové(přenos souborů), řídicí navazuje klient, datové navazuje server (okrem passive módu)
- příkazy – řízení přístupu, nastavení parametrů, výkonné příkazy
- TFTP

NFS (Network File System)

- Jeden ze sady protokolů TCP/IP, slouží pro transparentní sdílení souborů (uživatel/aplikace si neuvědomuje že se soubor nachází na vzdáleném počítači),

typicky se vzdálený soubor chová „tváří“ jako místní soubor a také se s ním tak pracuje

- Je použitelný na různých platformách
- Bezstavový protokol (v4 už je ale stavový), díky tomu je velmi robustní – to je důvod jeho úspěšnosti
- Využívá protokoly RPC (vzdálené volání procedur) a XDR (definuje jednotný způsob reprezentace přenášených dat nezávislý na konkrétní architektuře příjemce a odesílatele)
- mount server

HTTP (Hyper-Text Transfer Protocol)

- Protokol původně určený pro výměnu hypertextových dokumentů ve formátu HTML, v současné době užíván i pro přenos dalších informací – pomocí rozšíření MIME umí přenášet jakýkoli soubor (podobně jako SMTP)
- Existuje bezpečnější verze HTTPS, umožňuje přenášená data šifrovat
- Funguje systémem dotaz-odpověď, při zaslání více dotazů není možné rozpoznat zda spolu souvisí – HTTP je bezstavový protokol (nepříjemná vlastnost pro implementaci složitějších procesů přes HTTP) – proto byl rozšířen o HTTP cookies (umožňují uchovávat info o stavu na počítači uživatele)
- verze 0.9 – bez hlaviček, minimální možnosti
- verze 1.0 – rozšiřující hlavičky, podpora MIME
- verze 1.1 – virtuální servery, jedno spojení pro více přenosů (keep-alive), komprimace dat
- GET, HEAD, POST
- cookies
- cachování

HTML (Hyper-Text Markup Language)

- značkovací jazyk pro hypertext, definuje obsah, nikoliv vzhled
- CSS
- statické a dynamické HTML dokumenty – CGI, ISAPI, NSAPI, ASP, PHP
- skripty, Java, ActiveX

XML (eXtensible Markup Language)

- Značkovací jazyk vyvinut a standardizován konsorciem W3C, umožňuje snadné vytváření konkrétních značkovacích jazyků pro různé účely
- Určen především pro výměnu dat mezi aplikacemi a pro publikování dokumentů, umožňuje popsat strukturu dokumentu z hlediska věcného obsahu, nezabývá se sám o sobě vzhledem dokumentu, vzhled dokumentu se definuje připojeným stylem
- Pomocí různých stylů je možné provést transformaci do jiného typu dokumentu, nebo jiné XML struktury (výsledkem může být např. HTML, PostScript, . . .)

XSLT (eXtensible Style Sheet Language Transformations)

- Transformace sloužící pro převod dat ve formátu XML do lib. jiného požadovaného formátu (nejčastěji HTML, jiného XML, ale také PDF, či RTF, ...), struktura výstupu není definována přímo standardem – je závislá na procesoru XSLT (program který provede transformaci)
- K provedení transformace jsou třeba 2 soubory:
 - Soubor, který obsahuje zdrojová data, která budou transformována. Struktura tohoto souboru vyjma obecných vlastností XML není blíže specifikována.
 - Soubor, obsahující vzorec pro transformaci, napsaný v jazyce XSL