Architektura univerzálních procesorů

Architekturou procesoru rozumíme ideový návrh, tj. vytyčení principů jeho výstavby s přihlédnutím k potřebám programování, členění na nejdůležitější části a stanovení jejich vlastností atd. K charakteristice architektury mikroprocesoru postačí: **výčet registrů** a jejich funkcí, popis vnitřních a vnějších **sběrnic**, způsob **adresování** a **instrukční soubor**.

Registr je malé uložiště dat v mikroprocesoru s rychlým přístupem, které slouží jako pracovní paměť během výpočtů.

Zápisníková paměť – organizovaná jako paměť registrová. Slouží k ukládání mezivýsledku a dat při výpočtu procesoru

Zásobníková paměť – LIFO paměť. Princip zásobníkového adresování je vhodný pro vyčíslování složitých aritmetických a logických výrazů a při organizaci přerušení a volání podprogramů. (V těchto případech je nutno zaznamenat stavy přerušených programů a potom je v opačném pořadí obnovit.)

Sběrnice je soustava vodičů, která zajišťuje přenos informací mezi více než dvěma účastníky tak, že jeden informace vysílá a druhý je přijímá. Obecně dělíme sběrnici na 3 dílčí sběrnice:

- Datová
- Adresová
- Řídící

Architektury počítačů

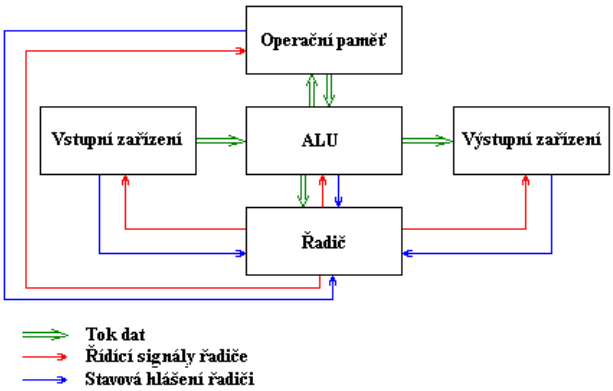
Von Neumannovo schéma počítače

John Von Neumann definoval v roce 1945 základní koncepci počítače, podle kterého se konstruují i dnešní počítače. Je postaveno na těchto principech:

- Počítač se skládá z
  - **Operační paměti –** slouží k uchování programu, dat a výsledků výpočtu. Jelikož přístup do hlavní paměti je časově náročný vznikla zásobníková paměť (skupina rychlých registrů) ke které se přistupuje sekvenčně jako na haldu.
  - **řídící jednotky (radič) –** Řídí činnost všech částí počítače pomocí řídících signálů.
  - **aritmetické jednotky (ALU) –** provádí aritmetické a logické operace. CPU je složeno z ALU + řadič.
  - **vstupní a výstupní jednotky**
- Struktura počítače je univerzální, počítač se programuje obsahem paměti
- Následující krok počítače je závislý na kroku předchozím
- Instrukce a data jsou v té samé paměti
- Paměť je rozdělena do buněk o stejné velikosti
- Program je tvořen posloupností instrukcí, které se vykonávají za sebou (lze provést podmíněný nebo nepodmíněný skok)
- Používá se dvojková číselná soustava

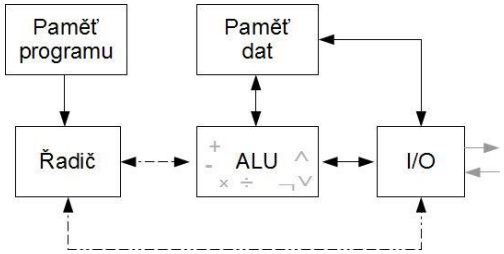
Výhody a nevýhody

- + rozdělení paměti na data a kód určuje programátor
- + jedna sběrnice – jednodušší výroba
- - společné uložení dat a kódu může mít za následek přepsání vlastního programu
- - jedna sběrnice tvoří úzké hrdlo



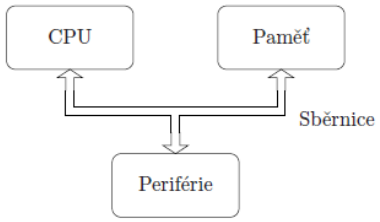
Harvardské schéma počítače

- Oddělení paměti pro data a program
- Základním nedostatkem obou koncepcí je sekvenční vykonávání instrukcí, které sice umožnuje snadnou implementaci systému, ale nepovoluje dnes tolik potřebné paralelní zpracování.

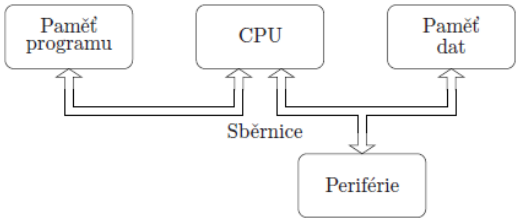


Výhody a nevýhody

- + program nemůže přepsat sám sebe
- + paměti mohou být vyrobeny odlišnými technologiemi
- + každá paměť může mít jinou velikost nejmenší adresovací jednotky
- + dvě sběrnice umožňují jednoduchý paralelizmus
- - dvě sběrnice kladou vyšší nároky na vývoj řídící jednotky, zvyšují náklady
- - nevyužitou část paměti dat nelze použít pro program a obráceně



Obrázek 2: Počítač podle von Neumanna



Obrázek 3: Harvardská architektura počítače

Procesory CISC a RISC

V dnešní době se ustálilo dělení počítačů do dvou základních kategorií podle typu použitého procesoru:

- **CISC** - počítač se složitým souborem instrukcí (Complex Instruction Set Computer)
- **RISC** - počítač s redukovaným souborem instrukcí (Reduced Instruction Set Computer)

CISC

- Počítač s úplnou instrukční sadou. Procesory obsahují množství instrukcí, kterých cílem bylo zjednodušit a vytvořit programy zabírající méně paměti. Důsledkem toho je architektura procesoru složitá.
- Procesory CISC jsou charakteristické velmi košatou instrukční sadou strojových instrukcí, instrukce mají proměnlivou délku i dobu vykonání a procesor obsahuje relativně nízký počet registrů. Paradoxně se tak může stát, že operace provedená složenou instrukcí (například násobení) může být nahrazena sledem jednodušších strojových instrukcí (sčítání a bitové posuvy), které mohou být ve výsledku vykonány rychleji, než hardwarově implementovaná složená varianta.

RISC

- Malý instrukční soubor
- V každém strojovém cyklu by měla být dokončena jedna instrukce
- Používá **zřetězené zpracování instrukcí**
- Celkový počet instrukcí a způsobu adresování je malý
- Data jsou vybírána a ukládána jen pomocí instrukcí LOAD s STORE
- Instrukce mají pevnou délku a jednotný formát
- Je použit vyšší počet registrů

Procesoru je třeba přenechat jen tu činnost, která je nezbytně nutná a převést další potřebné funkce do architektury počítače, programového vybavení a kompilátoru. Např. navýšení poctu registru souvisí s omezením komunikace s pamětí na dvě instrukce LOAD a STORE. Pokud ostatní instrukce nemohou používat paměťové operandy, je třeba mít v procesoru uloženo více dat. Další souvislost je patrná mezi zřetězeným zpracováním a formátem instrukcí. Jednotná délka instrukcí dovoluje rychlejší výběr instrukcí z paměti a tím zajišťuje lepší plnění fronty instrukcí (viz další kapitola). Jednotný formát pak zjednodušuje dekódování instrukcí.

Mezi nevýhody RISC architektury patří třeba nutný nárůst délky programu, tvořených omezeným počtem instrukcí a také díky jednotné délce všech instrukcí.

Principy urychlování činnosti procesorů

Urychlování procesorů se uskutečňuje jak z HW tak i ze SW hlediska. Zrychlení je možné dosáhnout např. zřetězením instrukcí, zvyšováním frekvence jádra, zdokonalováním technologie výroby, zvyšováním počtu jader, rychlejšími cache paměti a registry, paralelizací atd…

Zřetězené zpracování instrukcí

Provedení jedné instrukce musí projít vždy stejnými fázemi (viz tabulka). Pokud bude pro provedení jednoho elementárního úkonu potřeba jeden cyklus, můžeme provádění instrukce zobrazit v tabulce 4. Provedení jedné instrukce vyžaduje 6 cyklů a další instrukce se začne provádět, až se úplně dokončí všechny kroky předešlé instrukce. Když se jeden sekvenční obvod rozdělní na několik samostatných

obvodů (obrázek 2), tak aby jeden obvod odpovídal jedné fázi zpracování instrukce, mohly by se instrukce provádět tak jak je v tabulce 5 (princip výrobní linky). Mezi jednotlivé fáze zpracování však musí být pro mezivýsledky zaražen registr, který slouží jako předávací místo mezi po sobe jdoucími obvody.

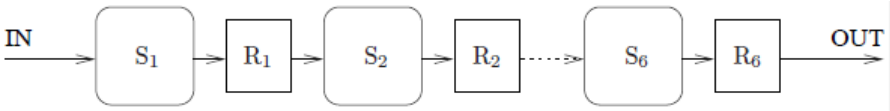
Aby uvedený princip zřetězení mel co největší přínos, musí být všechny fáze zpracování stejné časově náročné. Jinak je jasné, že nejpomalejší článek zřetězení bude brzdit všechny ostatní.

Krok		Význam
1.	**VI**	Výběr Instrukce
2.	**DE**	Dekódování
3.	**VA**	Výpočet Adresy
4.	**VO**	Výběr Operandu
5.	**PI**	Provedení Instrukce
6.	**UV**	Uložení Výsledku

Tabulka 3: Příklad možných kroků zpracování instrukce

	T_1	T_2	T_3	T_4	T_5	T_6	T_7	T_8	T_9	T_{10}	T_{11}	T_{12}	T_{13}
VI	I_1						I_2						...
DE		I_1						I_2					
VA			I_1						I_2				
VO				I_1						I_2			
PI					I_1						I_2		
UV						I_1						I_2	

Tabulka 4: Postup provádění instrukcí procesorem CISC



Obrázek 2: Zřetězené zpracování (v procesoru RISC)

	T_1	T_2	T_3	T_4	T_5	T_6	T_7	T_8	T_9	T_{10}	T_{11}	T_{12}
VI	I_1	I_2	I_3	I_4	I_5	I_6	I_7	...				
DE		I_1	I_2	I_3	I_4	I_5	I_6	I_7				
VA			I_1	I_2	I_3	I_4	I_5	I_6	I_7			
VO				I_1	I_2	I_3	I_4	I_5	I_6	I_7		
PI					I_1	I_2	I_3	I_4	I_5	I_6	I_7	
UV						I_1	I_2	I_3	I_4	I_5	I_6	I_7

Tabulka 5: Zřetězené provádění instrukcí procesorem RISC

Problémy zřetězení

Datové a strukturální hazardy

- **Datové hazardy** vznikají, když některá rozpracovaná instrukce potřebuje mít k dispozici data předchozí instrukce a ty ještě nejsou k dispozici.
    - **Řešení:** zřetězená jednotka je uzpůsobena svou konstrukcí nebo se tyto problémy řeší už v překladači, aby se podobným situacím zabránilo

- **Strukturální hazardy** vznikají např. při komunikaci po sběrnici při výběru operandu, ukládání výsledku nebo výběru instrukce. Na jednu sběrnici nelze povolit přístup více jednotkám současně, přístup na sběrnici je třeba koordinovat.

Problémy plnění fronty instrukcí

Pro optimální činnost zřetězeného zpracování je důležitá reakce na skokové instrukce. Problém se dá řešit podle typu skoku:

- **Nepodmíněný skok** – pokud se takový skok musí vypočítávat, je vhodné to řešit optimalizací na úrovni překladače (optimalizace pořadí strojových instrukcí). Snažíme se uspíšit výpočet adresy návěstí.
- **Podmíněný skok –** většinou známe adresu skoku, ale nevíme, zda se skok provede
    - Lepší je pokračovat ve zpracovávání sekvenčním způsobem, a pokud se skok neprovede, tak se rozpracovaná fronta instrukcí použije. V opačném případě se rozpracované instrukce ignorují a fronta se začne plnit znovu.
    - **Bit predikce skoku** - Instrukce skoku obsahuje jeden bit navíc, který říká, zda předpokládáme, že bude skok proveden
    - **Zpoždění skokové instrukce –** před skokovou instrukcí na prázdné místo dát několik instrukcí, které se skokovou instrukcí nesouvisí
    - **Zdvojení –** dvě paralelní fronty, když nastane podmíněný skok, začnou se vykonávat instrukce obou variant.

Základní vlastnosti monolitických počítačů

Pro řadu nenáročných aplikací se vyrábějí malé **počítače integrované v jednom pouzdře** – tvoří jeden celek **monolit**. Může se to také jmenovat jako jednočip, mikročip, mikrokontrolér a mikropočítač.

Pro monolitické počítače se převážně používá **harvardská architektura**, z důvodu oddělení paměti pro data a program. To umožnuje snadno konstruovat počítač, který má paměť dat i programu vyrobenou jinou technologií a navíc dovoluje mít odlišnou velikost nejmenší adresovací jednotky. Např. strojové instrukce procesoru mohou mít u každého procesoru jinou velikost a nejmenší adresní jednotka paměti se této velikosti přizpůsobuje.

Skládá se z **procesoru (CPU), paměti a periferií.**

Paměti monolitických počítačů

- **Pro data –** energeticky závislé RWM-RAM (paměti s náhodným přístupem určenou pro čtení i zápis). Buňky jsou realizovány jako klopné obvody.
- **Pro program –** ROM (obsah zachovávají i po odpojení napájení). EPROM a Flash EEPROM.

Organizace paměti dat

- **Střadačové (pracovní) registry -** Ukládají se do nich aktuálně zpracovávaná data a jsou nejčastějším operandem strojových instrukcí. A také se do nich nejčastěji ukládají výsledky operací.
- **Univerzální zápisníkové registry -** jsou jich desítky až stovky. Slouží pro ukládání nejčastěji používaných dat. Instrukční soubor obvykle dovoluje, aby se část strojových instrukcí prováděla přímo s těmito registry. Formát strojových instrukcí ovšem obvykle nedovoluje adresovat velký rozsah registru, proto se implementuje několik stejných skupin registru vedle sebe, s možností mezi skupinami přepínat - registrové banky.

- **Paměť dat RWM -** slouží pro ukládání rozsáhlejších nebo méně používaných dat. Instrukční soubor obvykle nedovoluje s obsahem této paměti přímo manipulovat, kromě instrukcí přesunových. Těmi se data přesunou např. do pracovního registru.

Zdroj synchronizace

Zdroj synchronizace vyžadují pro svou činnost počítače a ten vytváří časovou základnu pro provádění všech operací v počítači. Generátory umožňující zvolit si kmitočet pro řízení počítače:

- Krystal (křemenný výbrus)
- Keramický kondenzátor
- Obvod LC
- Obvod RC

RESET

Definuje počáteční stav počítače. Po provedení RESETU se u všech počítačů nastaví počáteční hodnota čítače instrukcí na nulu a připraví celý počítač na počáteční konfiguraci.

Ochrana proti přerušení

Na prvním místě většinou jde o ochranu mechanickou. Odolávat náhodným rázům, nebo i trvalým vibracím nebo elektromagnetickým vlivům z okolí. Pro odstranění chyb, které nastanou působením vnějších vlivů nebo chyby programátora, je v mikropočítačích implementován speciální obvod nazývaný WATCHDOG (provede pomocí vnitřního RESETu reinicilazaci mikropočítače) – patří k ochraně elektrické. Jde o samostatné běžící časovač, který svým přetečením, či podtečením, provede pomocí vnitřního RESETu reinicilazaci mikropočítače.

BROWN-OUT ochrana proti podpětí (-> reset). Pokud dojde k poklesu napěti pod určitou mez, automaticky se provádí RESET.

Integrované periferie

Obvody, které zajišťují komunikaci s okolím. Jejich kvalita a množství určují celkovou výkonnost systému.

Vstupní a výstupní brány

Nejjednodušší a nejčastěji používané rozhraní pro vstup a výstup informací je u mikropočítačů paralelní brána - port. Bývá obvykle organizována jako 4 nebo 8 jednobitové vývody, kde lze současně zapisovat i číst logické informace 0 a 1. U většiny bran lze jednotlivě nastavit, které bitové vývody budou sloužit jako vstupní a které jako výstupní. Slouží pro čtení hodnot dvoustavových snímačů a ovládání výkonových přepínačů, ovládání LED displeje.

Sériové rozhraní

Dovoluje efektivním způsobem přenášet data na relativně velké vzdálenosti při použití minimálního množství vodičů. Nevýhodou je nižší přenosová rychlost. Je potřeba kódování a dekódování dat. Pro přenos dat uvnitř zařízení mezi integrovanými obvody se používá standard I2C pro komunikaci mezi jednotlivými obvody.

Čítače a časovače

- **Čítač**

Čítač je N-bitový registr, který nejčastěji čítá vnější události. Obvykle lze nastavit, zda provádí inkrementaci při náběžné, nebo sestupné hrané vnějšího signálu.

- **Časovač**

Od čítače se příliš neliší. Není ale inkrementován vnějším signálem, ale přímo vnitřním hodinový signálem používaným pro řízení samotného mikropočítače. Lze tak podle přesnosti zdroje hodinového signálu zajistit řízení událostí a chování v reálném čase.

A/D převodníky

Fyzikální veličiny, které vstupují do mikropočítače, jsou většinou reprezentovány analogovou formou (napětím, proudem, nebo odporem). Pro zpracování počítačem však potřebujeme informaci v digitální (číselné) formě. K tomuto účelu slouží analogově–číslicové převodníky.

Typy:

- **Komparační A/D převodníky:** princip porovnávání měřené veličiny s referenční hodnotou, rozdělenou na několik hodnot v určitém poměru, napr. Odporovou děličkou.
- **A/D převodníky s D/A převodem –**
- **Integrační A/D převodník -**
- **Převodníky s časovacím RC článkem –**

D/A převodníky

Reálné prostředí, do kterého mikropočítače nasazujeme, vyžaduje nejen reakci na naměřené veličiny, ale i zásahy do tohoto prostředí. Tyto řídící podněty musí být často také v analogové formě. K tomuto účelu používáme číslicové analogové převodníky. Využívají se dva typy: **PWM (šířková modulace pulzů) a paralelní převodníky.**

Obvody reálného času

V mnoha aplikacích s použitím mikropočítačů je potřeba dodržovat přesnou časovou souvislost řízených událostí. Jde tedy o řízení v reálném case. Ne vždy ale taková posloupnost dostačuje a je nutno pro potřebu řízení udržovat skutečný čas, tedy hodiny, minuty, sekundy a připadne i zlomky sekund. Pro radu aplikací pak navíc musíme udržovat nejen denní čas, ale i kalendář, tedy týdny, měsíce a roky. Pro tyto účely slouží obvody hodin reálného casu – **RTC.** Pro udržení nepřetržité činnosti obvodu je potřeba záložní zdroj (v případě výpadku napětí).

Možnosti použití

- Kalkulačky
- Pokladny
- Automatické zavlažovací systémy
- Arduino, Raspberry pi
- Chytrá domácnost – senzory reagující na vnější podněty, řízení teploty atd.
- IOT

Struktura OS

Operační systém je prostředník mezi hardwarem a konkrétním programem, který uživatel používá.

K čemu je operační systém:

1. **Operační sytém jako stroj s rozšířenými možnostmi**

Funkcí operačního systému je prezentovat uživateli stroj s rozšířenými možnostmi. Nezajímá nás, jak funguje mechanika a jak ji mám naprogramovat. O to se postará OS, pomocí kterého využívám HW.

2. Operační systém jako správce prostředků

Úkolem OS je zajistit řádný a řízený přístup k procesoru, paměti, I/O zařízením mezi různými programy, které se o ně uchází. Hlavním úkolem je sledovat, kdo používá jaké zdroje, vyhovět žádostem o zdroje a řídit jejich používání a urovnávat konflikty mezi požadavky.

Co vše provádí OS

- Organizuje přístup ke zdrojům počítače a využívání zdrojů počítače (čas procesoru, přístup k datům na discích, přístup do paměti)
- Fyzicky zajišťuje vstup a výstup dat podle požadavků ostatních programů
- Komunikuje s uživatelem a na základě jeho pokynů vykonává požadované akce
- Reaguje na chybové stavy tak, aby tyto chyby nezpůsobily zásadní destrukci systému nebo poškození dat
- Spravuje komunikaci s periferiemi – klávesnice, myš atd.
- Eviduje využívání systémových zdrojů

Struktura OS (z čeho je OS tvořen):

- **Jádro (kernel) –** po zavedení do paměti řídí činnost počítače, poskytuje procesům služby a řeší správu prostředků a správu procesů
- **Ovladač I/O zařízení –** zvláštní podprogram pro ovládání konkrétního zařízení standartním způsobem. Strategie použití ovladačů umožňuje snadnou konfigurovatelnost technického vybavení
- **Příkazový procesor (shell) –** program, který umožňuje uživatelům zadávat příkazy ve speciálním obvykle jednoduchém jazyce
- **Podpůrné programy –** např. překladače a sestavující programy

Jádro (kernel)

Dělí se na dvě části:

- **Správa procesů** – řeší problematiku aktivování a deaktivování procesů podle jejich priority, resp. požadavků na prostředky
- **Správa prostředků** – zajišťuje činnost I/O zařízení, přiděluje paměť, případně procesory. Velmi důležitou částí je správa souborů – způsob ukládání souborů a přístupu k nim. Moderní OS zajišťují jednotný pohled na soubory a zařízení. Zařízení jsou považovány za soubory se speciálním jménem

Multitasking

Souběžné zpracování více úloh v teoreticky jednom okamžiku (je to tak, že se procesy rychle mezi sebou střídají)

- **Kooperativní multitasking** – přiděluje prováděným procesům procesor na takovou dobu, na jakou ji procesor potřebuje
- **Preemtivní multitasking** – vysoce výkonný, OS rozhoduje, komu přidělí jakou dobu procesor. Toto nastavení lze měnit prioritami

Přerušení – proces, během kterého je procesor nucen zaznamenat nějakou událost (např. stiskl klávesy), interrupt request (IRQ).

Prostředky přerušení – dovolují OS koordinovat paralelně probíhající operace – tím je umožněn paralelní běh uživatelských programů.

Životní cyklus procesu

- **Stav probíhající** – procesorový čas je přidělen procesu a ten je vykonáván
- **Stav čekající** – proces čeká na určitou událost, např. dokončení I/O operace
- **Stav připraven** – proces je připraven k vykonání a čeká na přidělení procesového času

Generické komponenty

Sadu vlastností, které musí mít každý operační systém:

- Správa procesorů
- Správa procesů (proces – činnost řízená programem)
- Správa hlavní paměti
- Správa vnitřní paměti
- Správa souborů
- Správa I/O systémů
- Správa vnější paměti
- Networking, distribuované systémy
- Systém ochran
- Interpret příkazů

Správa procesorů

- Sleduje procesor a stav procesů
- Rozhoduje, komu bude dána možnost užít procesor
- Přiděluje procesu prostředek (procesor)
- Požaduje vrácení procesoru

Správa procesů

- Vytváření a rušení procesů
- Potlačení a obnovení procesů
- Poskytnutí mechanismů pro synchronizaci procesů a pro komunikaci mezi procesy

Správa hlavní paměti

- Vede přehled kdo a kterou část paměti v daném okamžiku využívá
- Rozhoduje kterému procesu uspokojit jeho požadavek na prostor paměti
- Přiděluje a uvolňuje paměť podle potřeby
- Řídí virtuální paměť

Správa I/O systémů

Správce periferních zařízení má tyto funkce:

- Sleduje stav prostředků (periferních zařízení)
- Rozhoduje o efektivním způsobu přidělování periferních zařízení
- Přiřazuje periferní zařízení a zahajuje I/O operaci

- Požaduje navrácení periferního zařízení

Do správy I/O systému patří i správa vnější paměti (HDD)

Správa vnější paměti

- Správa volné paměti
- Přidělování paměti
- Plánování činnosti disku

Správa souborů

- Sleduje soubor, jeho umístění, užití, stav…
- Rozhoduje, komu budou soubory přiděleny, realizuje požadavky na ochranu informací uložených v souborech a realizuje operace přístupu k souborům
- Otevírá a uzavírá soubor

Distribuovaný systém

Kolekce procesorů nesdílejících ani fyzickou paměť, ani hodiny. Procesory distribuovaného systému jsou propojeny komunikační sítí. Komunikace je řízena protokoly.

Systém ochran

Mechanismus pro řízení přístupu k systémovým a uživatelským zdrojům. Tento systém je součástí všech vrstev OS.

- Rozlišuje mezi autorizovaným a neautorizovaným použitím
- Poskytuje prostředky pro své prosazení

Interpret příkazů

Program umožňující vykonávat příkazy pro:

- správu a vytváření procesů
- ovládání I/O zařízení – uživatelský program nesmí provádět I/O operace přímo, OS musí poskytovat prostředky k provádění I/O operací
- správa sekundární paměti
- správa hlavní paměti
- práce v síti

Struktura OS podle jádra

- monolitická jádra
- otevřené systémy
- mikrokernel

Monolitický OS

OS je na jednom místě. Aplikace používá dobře definované rozhraní systémových volání pro komunikaci s jádrem.

- **Výhody**
  - Dobrý výkon
  - Dobře pochopená koncepce
  - Jednoduché pro vývojáře jádra

    - Vysoká úroveň ochrany mezi aplikacemi
- **Nevýhody**
    - Žádná ochrana mezi komponentami jádra
    - Není jednoduše a bezpečně rozšiřitelné jádro
    - Celková struktura je ve výsledku komplikovaná, neexistuje hranice mezi moduly jádra

Otevřené systémy

Aplikace, knihovny i jádro jsou všechny v jednom adresovém prostoru. Příklady: MS-DOS, Win až do 98. Bývala běžná koncepce.

- Výhody
    - Velmi dobrý výkon
    - Velmi dobře rozšiřitelné
    - Výhodné pro jednouživatelské OS
- Nevýhody
    - Žádná ochrana mezi jádrem a aplikacemi
    - Nepříliš stabilní
    - Skládání rozšíření může vést k nepředvídatelnému chování

Mikrokernel OS

Chráněné jádro obsahuje pouze minimální množinu abstrakcí:

- Procesy a vlákna
- Vnitřní paměť
- Komunikace mezi procesy

Všechno ostatní jsou server-procesy běžící na uživatelské úrovní

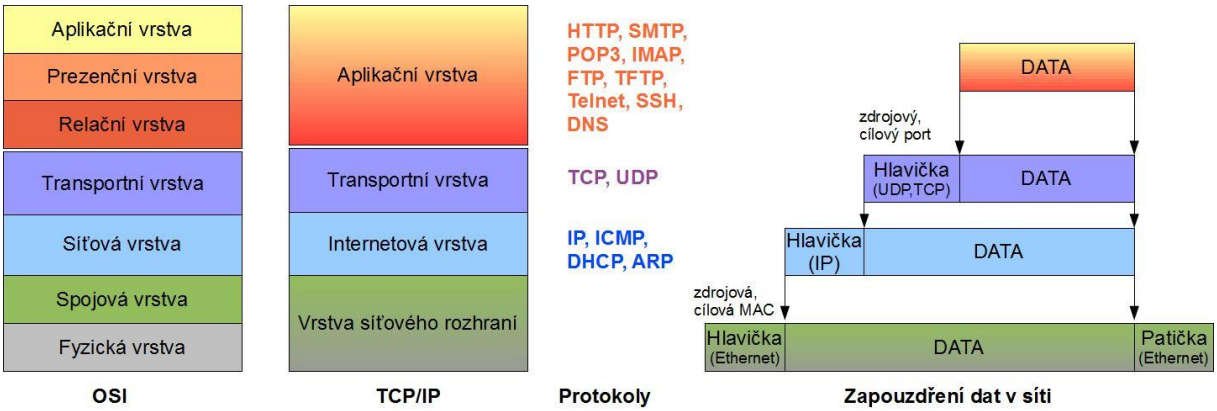
- Výhody
    - Přidáním server-procesu se rozšíří funkcionalita OS
    - Jádro nespecifikuje prostředí OS
    - Servery na uživatelské úrovni se nemusí zabývat hardwarem
    - Silná ochrana OS i sám proti sobě (části OS jsou oddělené servery)
    - Jednoduché rozšíření na distribuovaný nebo multiprocesorový systém
- Nevýhody
    - Výkon
    - Špatná minulost

Protokolová rodina TCP/IP

Rodina protokolu **TCP/IP** (Transmission Control Protocol/Internet Protocol) obsahuje sadu protokolů pro komunikaci v počítačové síti a je hlavním protokolem v celosvětové síti Internet. Komunikační protokol je množina pravidel, která určuje, jak vlastně probíhá komunikace mezi uzly v síti.

Architektura TCP/IP vychází z referenčního modelu ISO/OSI, což je takový pokus o standardizaci počítačových sítí. Obsahuje 7 vrstev, architektura TCP/IP z toho vychází, ale má pouze 4 vrstvy:

- Aplikační vrstva (zahrnuje Aplikační vrstvu, Prezentační vrstvu a Relační vrstvu OSI)
- Transportní vrstva (Transportní vrstva v OSI)
- Internetová (nebo také síťová) vrstva (Síťová vrstva v OSI)
- Vrstva síťového rozhraní (Spojová a fyzická vrstva v OSI)



Aplikační vrstva

Aplikační vrstva je nejvyšší vrstvou síťové architektury a obsahuje všechny protokoly, které poskytují uživatelům konkrétní aplikace. Účelem vrstvy je poskytnout aplikacím přístup ke komunikačnímu systému a umožnit tak jejich spolupráci.

Při odesílání dat se provádí encapsulace (zapouzdřování - zabalování) od nejvyšší vrstvy dolů. Probíhá to tak, že aplikace vezme data (aplikační vrstva), která chce zaslat jiné stanici a doplní je o aplikační hlavičku. Tato data zašle nižší vrstvě (transportní), která odesílaná data rozdělí na segmenty, zabalí a přidá TCP (nebo UDP) hlavičku a vytvoří TCP segment.

Protokoly aplikační vrstvy

SMTP - Internetový protokol určený pro přenos zpráv e-mailů mezi servery. Slouží k odesílání zpráv na server a k přenosu zpráv mezi servery. Funguje na portu TCP/25.

POP3, IMAP – Protokoly, které umožňují výběr pošty ze schránky (serveru). IMAP stahuje pouze hlavičky a obsah až v případě, že si ho chceme přečíst. Zprávy ze schránky nejsou mazány. IMAP funguje na portu TCP/143 (šifrované spojení TCP/993). POP3 stahuje zprávy ze schránky rovnou a ze schránky je maže. Funguje na portu TCP/110 (šifrovaně TCP/995).

FTP (File Transfer Protocol) – protokol určen pro přenos souborů. Pracuje na portech TCP/21(řízení) a TCP/20 (vlastní přenos dat). Umožňuje přihlašování, specifikaci formátu přenášeného souboru (znakově, binárně), výpis vzdáleného adresáře atd. Pracuje na principu klient-server.

SFTP (SSH File Transfer Protocol) – protokol pro bezpečný přenos souborů. Běží na portu TCP/22.

Telnet – Protokol, který dovoluje uživateli klientského terminálu připojit se ke vzdálené stanici či síťového uzlu a může mu zadávat příkazy. Využíván pro vzdálenou administraci. Pracuje na portu TCP/23

DNS (Domain Name Systém) – hierarchický systém doménových jmen, který je realizován servery DNS a protokolem stejného jména. Slouží k převodům doménových jmen na IP adresy uzlů. Používá porty TCP/53 i UDP/53.

TFTP (Trivial FTP) – jednoduchý protokol pro přenos souborů, obsahuje jen základní funkce FTP protokolu. Slouží např. k bootování bezdiskových počítačů ze sítě, kdy se celý přenosový protokol musí vejít do omezeného množství paměti, která je k dispozici na bez diskovém stroji. Používá portu UDP/69.

SSH (secure Shell) – protokol pro šifrovanou komunikaci pro vzdálenou administraci (šifrovaná náhrada Telnetu). Maká na portu TCP/22. Sklouží také ke kopírování souborů.

Transportní vrstva

Poskytuje mechanismus pro koncový přenos dat mezi dvěma stanicemi. Implementována je až v koncových zařízeních. Směrování je podle TCP hlavičky, která obsahuje **zdrojový a cílový port**. Tato vrstva rozdělí data z předešlé vrstvy na segmenty a doplní je o svojí hlavičku.

Protokoly transportní vrstvy

TCP (Transmission Control Protocol) – Použitím tohoto protokolu mohou uzly vytvořit mezi sebou spojení, pomocí kterého mohou přenášet data. Protokol garantuje spolehlivé doručování a doručování ve správném pořadí. Je to tzv. Spolehlivý protokol. Pro navázaní spojení se používá **třícestný handshake**. TCP ověřuje, zda přenesená data nebyla poškozená tak, že si před odesláním spočte kontrolní součet. Příjemce zasílá potvrzovací zprávy, které jsou potvrzením o úspěšně přijatých paketech.

UDP (User Datagram Protocol) – stejně jako TCP se stará o přenos dat, ale není spolehlivý. Nezaručuje správné pořadí paketů a ztraceného pakety se znovu neposílají. Používá se hlavně u streamu videa, VoIP.

Internetová vrstva

Zajišťuje síťovou adresaci, směřování a předávání datagramů. Poskytuje spojení mezi systémy, které spolu přímo nesousedí. Poskytuje funkce k zajištění přenosu dat přes několik propojených sítí. Všechny směrovače pracují na této vrstvě. Data z předešlé vrstvy se doplní o IP hlavičku a takto vznikne **IP paket**.

Protokoly internetové vrstvy

IP (Internet Protocol) – základní protokol pro přenos dat po síti. Data se po sítí posílají v paketech, které putují sítí zcela nezávisle na sobě. Na začátku spojení není třeba navazovat spojení., pakety se jen odešlou dál. Jednotlivé zařízení pak pakety posílají dál podle IP hlavičky, která obsahuje zdrojovou a cílovou **IP adresu**.

Každé síťové rozhraní komunikující prostřednictvím IP má přiřazeno jednoznačný identifikátor, tzv. IP adresu. V každém datagramu je pak uvedena IP adresa odesilatele i příjemce. Na základě IP adresy příjemce pak každý počítač na trase provádí rozhodnutí, jakým směrem paket odeslat, tzv. směrování (routing); na starosti to mají hlavně specializované stroje označované jako směrovače (routery).

ARP (Address Resolution Protocol) – Používá se k získání MAC (ethernetové adresy) sousedního stroje z jeho IP adresy. Používá se v situaci, kdy je třeba odeslat IP datagram na adresu ležící ve stejné podsíti jako odesilatel. Data se tedy mají poslat přímo adresátovi, u něhož však odesilatel zná pouze IP adresu. Pro odeslání prostřednictvím např. Ethernetu ale potřebuje znát cílovou ethernetovou adresu. Proto vysílající odešle ARP dotaz obsahující hledanou IP adresu. Tento dotaz se posílá jako broadcast všem. Dotaz nepřekročí hranice lokální sítě.

ICMP (Internet Control Message Protocol) – Protokol, který je používán k odesílání chybových zpráv, např. pro oznámení, že požadovaná služba není dostupná nebo že daný router nebo PC není dosažitelný. Příkladem je příkaz ping, který posílá „echo request“ a očekává příjem zprávy „Echo Reply“, tak zjistí, že je daný uzel dostupný.

Vrstva síťového rozhraní

Nejnižší vrstva umožňuje přístup k fyzickému přenosovému médiu. Je specifická pro každou síť v závislosti na její implementaci. Hlavním úkolem protokolu je navazování a ukončování spojení s komunikačním médiem a modulace signálu na signály používané přenosovým médiem. Data

z předešlé vrstvy jsou zabalena do **Ethernetového rámce**, který je doplněn o ethernetovou hlavičku, která obsahuje zdrojovou a cílovou **MAC adresu.** Rámec je to, co skutečně putuje sítí, vznikají až na fyzické vrstvě síťového rozhraní. Příkladem sítě je např. Ethernet

Ethernet

Jeden z typů lokálních sítí, který realizuje vrstvu síťového rozhraní. V lokálních sítích je nejrozšířenější. Výhodou je jeho jednoduchost. Jednotlivé stanice jsou v ethernetu identifikovány svými MAC adresami.

Využívá se sběrnicová topologie – sdílené médium, kde v každém okamžiku může vysílat jen jedna stanice. Pro přístup ke sdílenému médiu se používá metoda CSMA/CD – stanice, která potřebuje vysílat, naslouchá co se děje na přenosovém médiu. Pokud je v klidu, začne stanice vysílat. Může se stát, že dvě stanice začnou vysílat ve stejný okamžik a vznikne kolize. Stanice, která detekuje kolizi vyšle JAM signál, po kterém se všechny stanice odmlčí a později se pokusí o nové vysílání poté, co počkají náhodou dobu.

Metody síleného přístupu ke společnému kanálu

Kanál vs médium: více kanálů může sdílet jedno přenosové médium (multiplexovat) jednotlivé kanály na společnost přenosovou cestu).

V sítích je běžné, že více kanálů sdílí jedno fyzické přenosové médium – několik virtuálních cest. Přístup několika uživatelů ke společnému kanálu musí být řízen a to pomocí několika metod.

Společný kanál – kanál je sdílen všemi stanicemi a v jednom okamžiku smí vysílat jen jedna stanice. Existují různé přístupové metody, které rozhodují, která stanice smí vysílat a která musí čekat. Dělí se podle toho, zda dochází ke kolizím na:

- **Deterministické (bezkolizní)**
  - Algoritmus, který určuje, v jakém pořadí mohou stanice na kanál přistupovat
  - Je zajištěno, že na kanál nebude nikdy přistupovat více stanic současně
- **Nedeterministické (kolizní)**
  - V algoritmu hraje roli náhoda – náhodně volené časové prodlevy
  - O přístup na kanál se může pokoušet více stanic současně – vznikají kolize, které se musí řešit

Nedeterministické metody

Nevýhoda: U nedeterministických metod je možnost zablokování kanálu neustálými kolizemi.

Kolizní slot – udává kolik času se ztratí nevyužitím kanálu vlivem řešení kolize.

Aloha

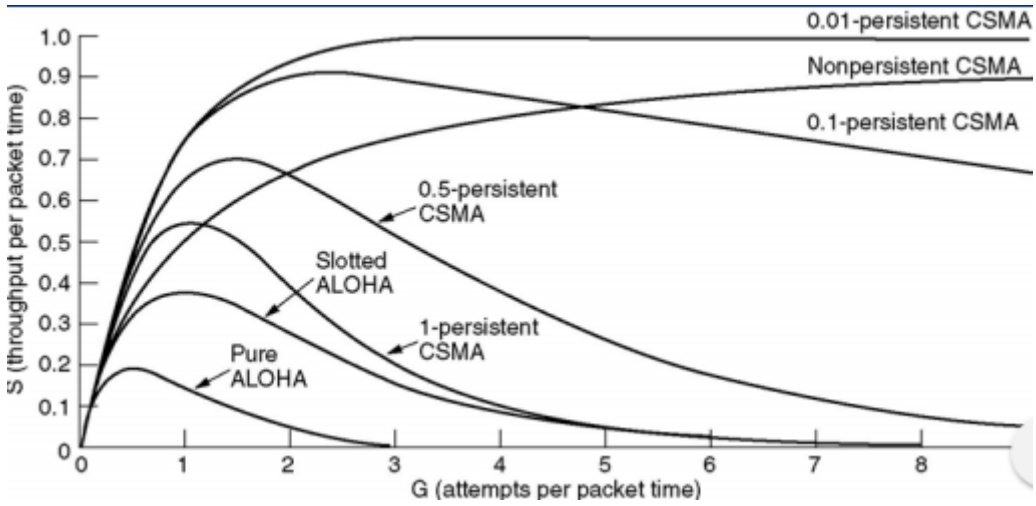
- Metoda vhodná pro kanál s řídkým provozem
- Netestuje se obsazenost média, rovnou se vysílá
- **Detekce kolice:** do časového limitu nepřijde potvrzení. Po vypršení limitu nastává opakování pokusu, před kterým je náhodné pozdržení
- **Použití:** pro rádiové a družicové sítě
- **Prostá aloha**
  - stanice vysílají bez ohledu na cokoliv

    - Kolizní slot: dvojnásobek doby vysílání rámce
- **Taktovaná aloha**
    - Vysílat se smí začít jen v okamžicích začátků časových úseků pro odeslání jednoho rámce
    - Kolizní slot je poloviční oproti prosté aloze a efektivita je dvojnásobná (37%)
    - Přenos bude úspěšný, pokud na začátku timeslotu začne vysílat právě jedna stanice, neúspěšný pokud stanic více
    - Malý vzrůst zátěže (jak pro prostou tak i pro taktovanou) může významně zvýšit počet opakování a snížit průchodnost kanálu – exponenciální závislost
- **Řízená aloha**
    - Řízená změna intenzity opakování podle okamžitého zatížení sítě (vyšší intenzita opakování => rychlejší předání rámce, při blížícím zablokování se však intenzita sníží)
    - Čím je zatíženější síť, tím je interval po kterém je možné opakovat pokus delší

CSMA (Carrier Sence Multiple Access)

- Skupina metod náhodného přístupu s příposlechem nosné, tj. využití znalostí o obsazení kanálu
- Podmínkou pro tuto metodu je dokonalá slyšitelnost stanic a malé zpoždění signálu, tyto podmínky splňuje LAN síť. Při nesplnění těchto podmínek má efektivitu horší než Aloha.
- **Naléhající CSMA (1-persistant)**
    - Před odesláním rámce se testuje stav kanálu, je-li kanál obsazen, odloží se vysílání na okamžik jeho uvolnění
    - Je tu ale riziko koliz stanic, které čekají na uvolnění kanálu – kanál se uvolní a začne najednou vysílat několik stanic -> jestliže kolize nastane, čeká se náhodnou dobu před dalším pokusem
- **Nenaléhající CSMA (non-persistant)**
    - Při detekci obsazeného kanálu se počká náhodnou dobu, pak opět test obsazení
    - Čekací doba se obvykle volí jako k-násobem doby průchodu signálu sdíleným médiem
- **P-naléhající CSMA (p-persistant)**
    - Při potřebě vysílání se počká na okamžik uvolnění kanálu (nebo je volný okamžitě)
    - S pravděpodobností p se začne vysílat, s pravděpodobností 1-p se počká 1 timeslot – toto se opakuje do úspěšného odeslání rámce
    - Pokud dojde ke kolizi, před dalším opakováním se počká náhodnou dobu
    - Volbou p lze nastavit optimální využití kanálu pro danou zátěž: p=1 jde o naléhající CSMA; pro velmi malá p značně narůstá průměrná doba doručení paketu

Efektivita metod CSMA a Aloha



CSMA/CD – CSMA with Collision Detection

- CSMA s detekcí kolize
- Při detekci kolize je pozastaveno vysílání a čeká se náhodnou dobu pro další pokus. Před vysíláním musí být médium v klidu po dobu jednoho kolizního slotu.
- V ethernetu se při detekci kolize vysílá kolizní JAM signál, aby kolizi rozpoznaly všechny kolidující stanice.
- Kanál se tak zbytečně nezaplňuje rámcem, který je stejně zkolidován

Deterministické metody

Nevýhoda: U deterministických metod je velká časová režie algoritmů (roste s počtem stanic). **Dělí se na**

- Centralizované řízení
- Distribuované řízení

Centralizované řízení

- Jedna stanice je vyhrazena jako řídící, ta přiděluje kapacitu kanálu ostatním stanicím. Výhodou je efektivita. Nevýhodou je, že se musí obětovat část kapacity kanálu pro komunikaci s masterem, další nevýhodou je závislost na řídící stanici

Centralizované řízení – přidělování na výzvu

- Nejstarší
- Stanice smí vysílat jen je-li k tomu vyzvána masterem
- **Cyklická výzva**
    - Master periodicky nabízí stanicím právo k vysílání
    - Vyzývaná stanice buď vyšle rámec, nebo odmítne výzvu, příp. neodpoví. Je to použitelné pro malé zpoždění na kanále. Rozumné pro vysoké využití kanálu většinou stanic
    - pro malé zatížení a velký počet stanic je neefektivní
- **Binární vyhledávání**
    - Při malém zatížení a velký počet stanic je efektivnější vyhledávání stanic, které jsou připravené vysílat pomocí binárního vyhledávání
    - Stanice se zorganizují do stromu podle jednotlivých bitů adres. Řídící stanice postupně vyzývá skupiny stanic (větve stromu) vysíláním hodnoty 0 nebo 1

příslušného bitu adresy. Stanice vyzvané skupiny, které chtějí vysílat, odpoví signálem na sdíleném kanále. Pokud stanice zjistí, že je jediná, která odpovídá, může začít vysílat. Jinak se výzva posune o jednu úroveň dolů ve stromu
    - Podmínkou je kanál, u kterého může stanice rozpoznat, zda vysílá jedna nebo více stanic

Centralizované řízení – přidělování na žádost

- **Ž**ádosti od stanic přicházejí k řídící stanici po vyhrazených kanálech. Řídící stanice pak přiděluje na základě žádostí kanál (typicky na společném downstream kanále).
- Použití v rádiových sítích. Pro žádost se často používá vyhrazeného nízko rychlostního podkanálu časového multiplexu.

Distribuované řízení

- Nezávislé na řídící stanici
- Složitější implementace
- **Rezervace kanálu**
    - Master periodicky generuje rezervační rámec, který má tolik slotů, kolik je stanic a stanice se do svého slotu může umístit požadavek na vysílání
    - V přenosovém kanálu je vyhrazený přenosový rámec, ve kterém si stanice rezervují přidělení kanálu na vysílání. Takto si každá stanice zapíše do rámce svůj požadavek na vysílání. Po ukončení rezervačního rámce každá stanice ví o všech stanicích, které budou chtít vysílat a jejich rezervacích. Vysílání následně probíhá v pořadí, jaké určuje rezervační rámec.
    - Neefektivní pro velký počet stanic na rozlehlé sítí s malou zátěží
- **Binární vyhledávání**
    - Stanice, které chtějí vysílat, postupně posílají bity své adresy.
    - Jakmile stanice vysílá 0, ale čte od jiné stanice 1, má ta stanice vyšší prioritu a stanice se musí odmlknout
    - Stanice, která úspěšně odešle celou svou adresu, může vyslat jeden rámce.
    - Vhodnou adresací lze řešit prioritu stanic
    -

	0 1 2 3
0 0 1 0	0 – – –
0 1 0 0	0 – – –
1 0 0 1	1 0 0 –
1 0 1 0	1 0 1 0
Result	1 0 1 0

- **Logický kruh**
    - Stanice tvoří kruh. Každá stanice zná svou adresu a adresu svého následovníka, takže mu může předat právo na vysílání (token). Ten mezi nimi pořád dokola putuje
    - Problémem je přidávání stanic za provozu

Problémy směrování v počítačových sítích

Směrování je technika, která slouží k propojení jednotlivých sítí. Zařízení, které se stará o směrování se nazývá **router,** který přeposílá komunikaci z jedné sítě do jiné. Routery propojují jednotlivé sítě, takže pokud chce nějaké zařízení komunikovat s jiným zařízením v jiné síti, pošle data na výchozí bránu – na svůj router, který se postará o odeslání dat do správné sítě (do jiného routeru, který se již postará o správné doručení).

Délka cesty mezi zařízeními se počítá podle počtu **hopů**, což je každý přechod ze zařízení na zařízení (počet routerů v cestě + 1).

Každý router se rozhoduje sám za sebe, kam pošle paket, který přijal. Závisí to na obsahu své vlastní **směrovací tabulky**, která obsahuje nejlepší cesty k cílům.

Směrovací tabulka

Datová struktura, podle které se rozhoduje, co udělat s přijatým packetem. Formát: *cílová adresa/maska; Výstupní rozhraní; Brána(next hop); Metrika*.

- Brána – IP adresa nejbližšího routeru, na který mají být pakety odeslány
- Rozhraní – označení síťového rozhraní routeru
- Metrika – vyjádření ceny při použití dané trasy. Obvykle je to počet **hopů**. Používá se pro výběr nejvhodnější cesty.

Jestliže se pro danou cílovou adresu nenajde příslušný záznam v tabulce, bude použita **výchozí cesta**, která je definovaná pro každý router a která obvykle směruje na nadřazený prvek.

Routovací protokoly

Routovací protokoly slouží ke směrování, určuje nejlepší cestu k cíli a posílá routovací informace dalším routerům. Dělí se na:

- **Statické routování** – ručně zadané adresy (záznamy v routovací tabulce). Změny v topologické síti se musí provést ručně.
- **Dynamické routování** – dva základní typy:
    - **Distance-vector**
        - routery udržují routovací tabulku s informací o vzdálenosti do dané sítě, periodicky svou routovací tabulku zasílají svým sousedům, ti si upraví svoji tabulku podle nových dat a tu opět odešlou dál. Pro výpočet nejlepší cesty se používá jedna (počet hopů) nebo více metrik (propustnost linky, zpoždění)
        - Směrovače neznají topologii sítě
        - Reprezentantem je RIP protokol
    - **Link-state**
        - routery udržují komplexní databázi síťové topologie, vyměňují si LSA, které jsou vyvolány nějakou událostí v síti. Metrika je komplexní, nejlepší cesta se počítá pomocí Dijkstrova algoritmu.
        - Směrování na základě stavu jednotlivých linek. Směrovače znají topologii celé sítě a ceny těchto linek. Každý směrovač počítá strom nejkratších cest ke všem směrovačům.
        - Neustálé sledování stavu linek zajišťuje komunikace mezi směrovači pomocí LSA Hello zpráv. Při změně okamžitě šíří informaci o stavu svého okolí všem ostatním. Spotřebovává víc pásma, hlavně na začátku zasílá množství LSP
        - Rychlejší konvergence – čas potřebný k tomu, aby měly všechny routery kompletní aktuální informace o topologii.
        - Reprezentant je OSPF protokol

Dynamické protokoly dále dělíme podle toho, zda jsou určeny pro nasazení uvnitř lokální sítě nebo fungují napříč sítěmi:

- **IGP** – routuje uvnitř

- **EGP** – routuje mezi sítěmi

Konkrétní routovací metody

- **RIP**
  - Jednoduchý na konfiguraci a funguje všude
  - Pro malé a střední sítě
  - RIP1 nepodporuje VLSM
  - Plýtvá pásmem – velká režijní komunikace
  - Pomalá konvergence
  - Hloupá metrika – jen počet hopů
  - Posílá celou routovací tabulku svým sousedům
  - Omezení maximálně na 15 hopů
  - Definují se pouze sítě, které jsou přímo na tomto routeru, ostatní se naučí pomocí updatů
- **OSPF**
  - Hierarchický systém – sítě jsou rozdělené na oblasti (area). LSA se říší pouze uvnitř dané oblasti a výpočet stromu se spouští také pro každou oblast zvlášť
  - Směrovač nejprve vypočte strom nejkratších cest a až pak z něj vytvoří směrovací tabulku
  - Šíří i masky podsítí, takže ji lze využít i v sítí s VLSM adresací
  - Vhodné pro rozsáhlé sítě

Switch a směrování

Switche pracují na druhé vrstvě TCP/IP a směrují data pomocí MAC adres v rámci lokální sítě. Pokud switch daný cíl nezná, odešle rámec na všechny své rozhraní – broadcast. To může vést k problému, pokud v sítí existuje cyklus a je vyslán broadcast, může se stát, že se se broadcast bude množit. To řeší tzv. **Spannig tree protocol**, který vytváří v sítí strom a odstraňuje tím smyčky. Dělá se to tak, že se odpojí redundantní spoje. Protokol dál dokáže odpojené spoje znovu zapojit, pokud dojde k přerušení aktivní cesty.

Adresování v IP

Základním protokolem současných sítí je protokol IP. V současnosti se používá verze 4 a nastupuje verze 6. Protokol IP zajišťuje

- Logické adresování sítí a koncových zařízení v nich
- Prostředky pro doručování paketů mezi zařízeními

Každé síťové rozhraní má samostatnou IP adresu. Zařízení má tolik adres, kolika rozhraními komunikuje. IP adresa je tvořena 4 oktety, které jsou oddělené tečkou. Jeden oktet má 8 bitů. Celá adresa IP verze 4 má tedy 32 bitů. IP adresa verze 6 má 128 bitů.

Každá adresa má dvě části:

- **Identifikátor sítě** – kolik prvních bitů tvoří adresu sítě, zjistíme pomocí masky sítě, která je v podobě např. 255.0.0.0 (první oktet je adresa sítě, protože tam jsou jedničky)
- **Identifikátor počítače v dané síti**

Pro směrování v routerech není identifikátor počítače zajímavý, směrování se provádí jen na základě identifikátoru sítě. Směrování v IPv4 je řízeno adresami cílových sítí.

Jak zjistit z IP adresy síťovou část

- **Podle třídy adres (A, B, C a D pro multicast, E je rezervováno)**
  - A: první oktet je síť
  - B: první dva oktety jsou síť
  - C: první tři oktety jsou síť, poslední oktet identifikuje zařízení v dané síti
- **Pomocí síťové masky –** od používání tříd se již ustoupilo a začalo se používat adresování CIDR
  - **CIDR**
    - dělí síť na podsítě. V komunikaci se používá vždy IP adresa spolu s maskou, podle které se pozná adresa síťe. IP adresa je rozdělena na:
      - Identifikátor sítě
      - Identifikátor podsítě
      - Identifikátor stanice
    - Jedna síť se tedy může rozdělit na více podsítí o stejné velikosti
  - **VLSM –** slouží k vytváření podsítí proměnlivé délky tak, aby se zbytečně neplýtvalo adresami, kde např. pro podsíť mezi routery stačí jen 4 adresy (2 pro interface, 1 pro broadcast, 1 pro označení sítě) a síť má masku /30.

IP adresy se dělí na:

- Unicast – adresa jednoho konkrétního zařízení
- Multicast – adresa více počítačů najednou
- Broadcast – adresa pro všechny počítače v rámci sítě – 255.255.255.255
- Loopback – adresa sama na sebe 127.0.0.1
- Některé rozsahy adres sou tzv. privátní, např. 192.168.0.0 – 192.168.255.255 a slouží pouze k adresování v lokální privátní sítí, která není viditelná z vnějšku.

IPv6

- Nastupující protokol, který nabízí ohromné množství IP adres
- IPv4 adresy dochází
- 128 bitová délka
- Skládá se ze dvou logických částí: 64 bitového síťového prefixu a 64 bitové části hosta (vytvářená automaticky na základě MAC zařízení)
- Příklad: 2001:0db8:85a3:08d3:1319:8a2e:0370:7334

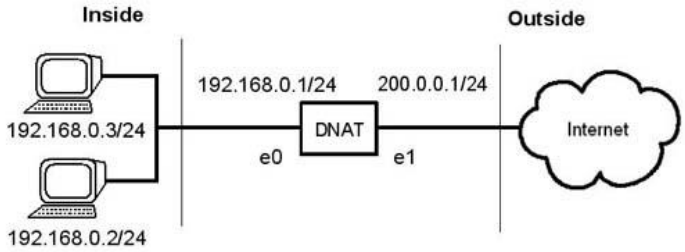
Překlad adres (NAT)

Network Address Translation (NAT)

- Překlad síťových adres
- Způsob úpravy síťového provozu procházejícího přes router přepisem zdrojové nebo cílové IP adresy, případě i čísel portů.
- Většinou se používá pro přístup více počítačů z lokální sítě ven do internetu prostřednictvím jedné veřejné IP adresy
- Pro přístup do internetu je potřeba unikátní veřejná IP adresa, kterých je omezené množství. To bylo jedním z důvodů, proč vzniklo NAT, díky kterému se celé rozsáhlé sítě s velkým počtem počítačů mohou schovat za jedinou unikátní IP adresu
- NAT tak představuje hranici mezi internetem WAN a LAN sítí.
- Pro překlad se využívají **překladové tabulky** – slouží k poznání zdroje z lokální sítě při odpovědi

Výhody NAT

- **Šetří adresní prostor IP adres** – celá jedna velká síť se chová za jednu veřejnou IP adresu
- **Zvyšuje bezpečnost vnitřní sítě** – Skryje vnitřní adresy. NAT vytvoří jednoduchý firewall a žádný počítač nemůže přímo kontaktovat počítač na vnitřní síti, pokud to předtím tento počítač neudělá sám.



NAT může fungovat různými způsoby

- **Dynamický NAT**
  - Udržuje se seznam globálních adres (NAT Pool), na které lze překládat lokální adresy
  - Umožňuje zpřístupnění vnější sítě mnoha stanicím jen přes několik globálních adres
  - Postup:
    - Zařízení se pokouší připojit na nějaký vnější zdroj
    - NAT prvek přijme paket
    - Pokud ještě počítač nemá přidělenou nějakou globální adresu, přidělí se mu nějaká z poolu a vytvoří se příslušný záznam v překladové tabulce
    - NAT vymění adresu v IP hlavičce podle té překladové tabulky a pošle paket ven
    - Když jde odpověď na požadavek zpět, NAT prvek zkontroluje cílovou adresu a z překladové tabulky zjistí, komu patří v lokální sítí. Vymění se IP a pošle se paket na daný počítač.
- **Statický NAT**
  - NAT prvek provádí statický překlad jedné IP adresy vždy na tutéž IP adresu
  - Užitečné, pokud potřebujeme sdílet nějaké zařízení z vnější sítě nebo při přechodu na jiný adresní rozsah.
  - Překladová tabulka je statická
- **PAT**
  - Více lokálních IP se převádí jen na jednu globální adresu, ale na různé porty
  - Funguje to tak, že se při překladu přidělí danému zařízení vygenerovaný ještě nepoužitý port, podle kterého se pak při odpovědi pozná, komu má daná odpověď jít.

Bezpečnost počítačových sítí s TCP/IP

Bezpečnost je proces, který se snaží zabránit zneužití systémů neoprávněnými osobami ať už ve smyslu vyřazení z funkce nebo odcizení či modifikace citlivých dat. Zajištění bezpečnosti je neustálý proces, který zahrnuje nejen nějaké technické opatření, ale i stanovení bezpečností politiky, které musí uživatelé dodržovat. Implementace bezpečnostní politiky je vždy pro uživatele omezující. Je třeba najít kompromis mezi pohodlím uživatelů a bezpečnosti. Bezpečnost zahrnuje zabezpečení jak síťové infrastruktury, tak i zabezpečí OS.

Základní pojmy

- **Utajení –** přenos dat takovým způsobem, že cizí posluchač naslouchající na přenosovém kanále přenášeným datům nerozumí
- **Autentizace –** dává příjemci jistotu, že odesílatel dat je skutečně tím, za koho se vydává
- **Integrita dat –** dává příjemci jistotu, že data nebyla na cestě nikým modifikována
- **Nepopíratelnost –** mechanismus zajišťující, že zdroj dat nemůže popřít, že jistá konkrétní data odeslal

Útoky

Útoky na počítačové sítě mohou mít velmi různorodý charakter. Rozlišujeme pokusy o průnik, kdy se útočník dostává k zařízením a útoky, kdy se útočník pokouší úplně zničit, případně modifikovat chování sítě nebo serveru.

Typy útoků

- Odposlech při přenosu
- Falšování identity (Man in the Middle)
- Programové útoky – viry, trojské koně
- Buffer overflow
- Přetížení či zahlcení zdrojů (DoS, DDoS)
- Získávání soukromých informací
- Podvržení informace

Obrana

- Znemožnění odposlechu pomocí **šifrování**
- blokování útoků – firewall
- znemožnění přístupu – VLAN, VPN

Denial of Service (DoS) útoky

Cílem útoku je zhroucení infrastruktury a omezení funkčnosti, tak aby bylo omezeno poskytování dosavadních služeb uživatelům. Cílem útočníka je vyčerpání systémových prostředků síťového prvku nebo serveru, typicky jsou to paměť, procesorový čas nebo šířka pásma.

DoS útoky zpravidla přicházejí z podvržených zdrojových adres, aby útočníci obešly případné paketové filtry (source IP spoofing). Všechny typy se vyznačují několika specifickými charakteristikami

- zaplavení provozu na sítí náhodnými daty, které zabraňují protékání skutečných dat
- zabránění nebo přerušení konkrétnímu uživateli v přístupu ke službě
- Extrémní vytížení CPU severu

Varianty

- DoS pomocí chyb v implementaci IP
    - **PingOfDeath** - Odeslání příliš velkého paketu pomocí ping
    - **Teardrops** – využívá chyby při skládání fragmentovaných paketů (posílá nekorektní pakety)
- **SYN flooding** – útočník zahájí navázaní TCP spojení (pošle paket SYN). Cíl potvrdí a alokuje pro otevírané spojení zdroje. Útočník ale nedokončí navázaní spojení, místo toho zahajuje otevírání dalších spojení.
- **Smurf** – zahlcení cíle ICMP pakety (ping), jejich zpracování mívá někdy přednost před běžným provozem

- **DDoS (Distributed Denial of Service)** – DoS útok vedený souběžně z mnoha stanic, které jsou infikovány virem. V určitý čas útočník vzbudí tyto nakažené zombie PC a pošle je současně na cíl. Obtížně se blokuje, protože zdrojů může být příliš mnoho

Detekce útoků a obrana proti únikům

Pro detekci útoků a průniků dnes existuje celá řada systémů, obecně nazývaná **Intrusion Detection Systems (IDS) – systém pro odhalení průniku**. Tyto systémy neustále sledují provoz na síti a na základě podezřelých vzorů chování vyhodnocují stavy, které mohou s jistou pravděpodobností identifikovat nějaký typ útoku. Způsob reakce na oznámené riziko je v IDS již ponechán na administrátorovi. U systému typu UPS (Preventing Systems) je již automaticky provedeno opatření, aby byl další provoz od útočníka odfiltrován.

Zabezpečení na jednotlivých vrstvách OSI-RM

- L2
    - Hop-by-hop, neefektivní
    - Layer 2 tunneling protocol (L2TP), point to point tulelling protocol (PPTP)
- L3 - IPSec
- L4 - Secure Sockets Layer (SSL)
- L7 - Řeší jednotlivé aplikace

Filtrace provozu

Zabezpečení operačních systémů a na nich běžícího software před útoky nebo neoprávněným přístupem v prostředí počítačové sítě se nejjednodušeji řeší vhodnou filtrací nežádoucího provozu. Filtrace může probíhat buďto na úrovni inspekce jednotlivých paketů za použití tzv. **paketových filtrů**, výsledkem jejichž vyhodnocení je vždy propuštění nebo zahození kontrolovaného paketu. Hovoříme zde o **filtraci bezestavové**, jelikož jednotlivé pakety jsou kontrolovány nezávisle na sobě, aniž by se filtrující zařízení (typicky směrovač) snažil rekonstruovat datový proud nesený pakety, které k sobě logicky patří. Bezestavová filtrace se tedy děje pouze na základě dat obsažených v paketu, nikoli podle „stavu“ logického spojení zprostředkovávaného jednotlivými pakety.

Náročnější variantou filtrace je **filtrace stavová**, při které si filtrující zařízení z procházejících paketů rekonstruuje obsah jednotlivých datových toků. Stavová filtrace dovoluje filtrovat na základě chování aplikačních protokolů na 7. vrstvě OSI RM, avšak platí za to omezenou škálovatelností, kdy pro každé jednotlivé spojení je třeba udržovat v paměti jeho stav a procesor musí nahlížet nejen do hlaviček, ale i do aplikačních dat.

Paketové filtry

Nejjednodušší a nejstarší forma firewallování, která spočívá v tom, že pravidla přesně uvádějí, z jaké adresy a portu na jakou adresu a port může být doručen procházející paket, tj. kontrola se provádí na třetí a čtvrté vrstvě modelu síťové komunikace OSI.

- **Výhody**
    - Vysoká rychlost zpracování, vhodné tam, kde je nutný vysokorychlostní přenos velkých množství dat
- **Nevýhody**
    - Nízká úroveň kontroly procházejících spojení, které zejména u složitějších protokolů (FTP, streaming) nejen nedostačuje ke kontrole vlastního spojení, ale pro umožnění

takového spojení vyžaduje otevřít i porty a směry spojení, které mohou být využity jinými protokoly, než bezpečnostní správce zamýšlel povolit.

- můžeme kontrolovat pouze korektnost dat zcela obsažených v jednom paketu, nikoli tedy nebezpečná data rozložená do více samostatných paketů

ACL

Mezi typické představitele paketových filtrů patří ACL (Access Control LIsts). Je to v podstatě filtr umístěný na některém rozhraní aktivního prvku (router, L3 přepínač). ACL je sekvence pravidel zakazujících nebo povolujících průchod paketů vyhovujícím kritériím definovaným daným pravidlem. Při průchodu paketu rozhraním, na němž je ACL aplikován, se postupně odshora procházejí jednotlivé položky ACL, až se narazí na první položku, jejímž kritériím zkoumaný paket vyhovuje. Podle toho, zda je položka definována jako příkaz k propuštění vyhovujícího paketu nebo k jeho zahození se pak paket propustí nebo zlikviduje. Na konci ACL typicky bývá implicitní položka přikazující „zahodit vše“, vlivem čehož ACL respektují filosofii „co není explicitně povoleno, je zakázáno“.

Stavový firewall

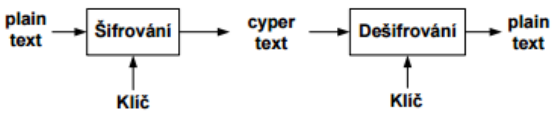
Stavovou filtraci na základě nedovolených aktivit detekovaných v datových tocích aplikačních protokolů realizuje zpravidla firewall. **Firewall** je síťové zařízení, které slouží k řízení a zabezpečování síťového provozu mezi sítěmi s různou úrovní důvěryhodnosti a zabezpečení. Zjednodušeně se dá říct, že slouží jako kontrolní bod, který definuje pravidla pro komunikaci mezi sítěmi, které od sebe odděluje. Firewall odděluje důvěryhodnou a nedůvěryhodnou část sítě, zkoumá datové toky mezi nimi a podle nakonfigurovaných pravidel je dovoluje nebo naopak zakazuje. Implementace firewallu může být buďto „hardwarová“ (specializované zařízení) nebo „softwarová“ s použitím vhodného operačního systému (Linux, NetBSD, …).

Firewally se často implementují ne pouze se dvěma rozhraními jako na obr. 1.33, ale s dalším třetím rozhraním, ke kterému je připojena tzv „**demilitarizovaná zóna**“ (DMZ). V DMZ se typicky vyskytují servery, které mají být přístupny jak z vnitřní sítě, tak z Internetu. Tyto servery (tzv. „bastillon hosts“) mají řádně zabezpečený operační systém, aby nemohlo dojít k jejich napadení. Na servery v DMZ smějí přistupovat jak uživatelé z vnitřní sítě, tak uživatelé z Internetu. Přímý přístup uživatelů z Internetu do vnitřní sítě je však zakázán.

Stavový firewall (též stavový paketový filtr, anglicky stateful firewall) je označení pro takový firewall, který podporuje SPI (anglicky Stateful packet inspection), což znamená, že je schopen sledovat a udržovat všechny navázané TCP/UDP relace. Stavový firewall pracuje na transportní vrstvě referenčního modelu ISO/OSI.

Šifrování a autentizace

Kryptografickým systémem rozumíme systém, který na straně odesílatele dat šifruje zprávu, přenáší ji zašifrovanou a na straně příjemce původní zprávu dešifruje. K šifrování i dešifrování se využívá nějaký veřejný algoritmus, jenž je však parametrizován pomocí klíčů, které jediné musí zůstat utajeny. Podmínkou je pouze to, by bylo dostatek klíčů, aby útočník nebyl schopen jednoduše vyzkoušet všechny z nich.



Obr. 1.29 – Šifrování a dešifrování informací během přenosu

Slovo **šifra** označuje algoritmus, který převádí čitelnou zprávu na její nečitelnou podobu. **Klíč** je tajná informace, bez níž nelže šifrovaný text přečíst. **Hašovací funkce** je způsob, jak z celého textu vytvořit krátký řetězec, který s velmi velkou pravděpodobností identifikuje nezměněný text.

Základní typy šifrování

S ohledem na způsob zacházení s klíči rozlišujeme **symetrické šifrování** a **asymetrické šifrování.**

Symetrické šifrování

- Je použit sdílený klíč, totožný na vysílači i přijímači.
- Symetrické algoritmy: DES, 3DES, AES
- Problémem je distribuce klíčů – přenos klíče je nešifrovaný

Symetrické šifrování – autentizace

V symetrickém systému založeném na sdíleném tajemství (hesle) mezi vysílačem a přijímačem můžeme uživatele autentizovat jednoduše tak, že jeho uživatelské jméno zašifrujeme klíčem a na přijímači je opět dešifrujeme a porovnáme se seznamem autorizovaných uživatelů.

Jestliže chceme pouze ověřit identitu odesílatele, aniž bychom k tomu potřebovali seznam uživatelských jmen na přijímači, můžeme postupovat tak, že na vysílači nejprve ke jménu uživatele připojíme jeho otisk (hash) a celou zprávu poté zašifrujeme. Přijímač, který zprávu jako celek dešifruje, pak může smysluplnost dešifrované informace ověřit tak, že z dešifrovaného uživatelského jména stejnou hash funkcí jako předtím (např. MD5) vysílač vypočte otisk a srovná jeho shodu s dešifrovaným otiskem.

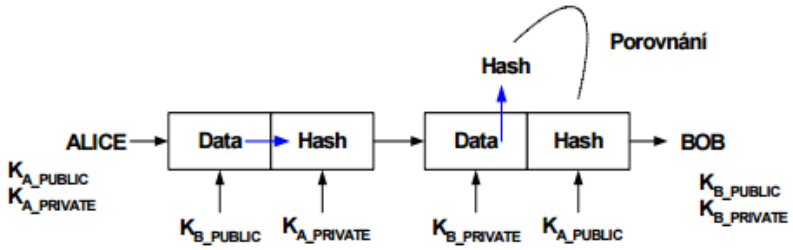
Společně s autentizací odesílatele se zpravidla implementuje i **zajištění integrity přenášených dat**. Zajištění integrity mezi uživateli sdílejícími tajný klíč se realizuje tak, že odesílatel v paměti za zprávu připojí sdílený tajný klíč a z celého bloku **[zpráva+sdílený tajný klíč]** vypočte otisk. Poté vyšle původní zprávu a otisk celé dvojice. Přijímač pak za došlou zprávu v paměti připojí sdílený klíč a stejnou hash funkcí jako vysílač vypočte otisk. Ten poté srovná s otiskem, který spolu se zprávou vyslal vysílač.

Asymetrické šifrování

- Klíče se generují jako doplňující se pár – **veřejný** a **soukromý klíč.** Jeden z těchto klíčů je vždy použit pro šifrování, druhý pro dešifrování (je jedno který)
- Generování páru si každý uživatel může uskutečnit lokálně, privátní klíč si bezpečně skrýt a veřejný klíč zveřejnit.
- Algoritmy (RSA) jsou mnohem složitější, náročnější na výpočty a pomalejší
- Používá se pro předávání klíčů pro symetrické šifrování.
- Podvržení veřejně distribuovaných veřejných klíčů se obvykle zabraňuje s použitím **certifikační autority**, která svým privátním klíčem podepisuje certifikáty jednotlivých uživatelů. Certifikátem rozumíme blok dat tvořený jednak veřejným klíčem a jednak identifikací vlastníka veřejného klíče. Pravost podpisu lze ověřit pomocí veřejného klíče certifikační autority, která certifikát podepsala. Jednotlivým uživatelům tak stačí spolehlivým způsobem získat jediný veřejný klíč certifikační autority, který mu již zajistí ověření pravosti veřejných klíčů ostatních uživatelů.

Asymetrické šifrování - autentizace

V asymetrickém systému se autentizuje poněkud odlišným způsobem, na kterém jsou mj. založeny i elektronické podpisy. Princip je vidět z obrázku:



Obr. 1.31 – Autentizace v asymetrickém systému

Uživatel Alice chce poslat zprávu uživateli Bob. Z dat nejprve vypočte otisk a ten zašifruje svým soukromým klíčem KA_PRIVATE. Data zprávy zašifruje veřejným klíčem Boba, který jako jediný může zprávu dešifrovat svým soukromým klíčem KB_PRIVATE. Mimo to Bob prověří, zda zprávu vyslala Alice tak, si z dešifrované přijaté zprávy sám spočítá otisk a dešifrovaný otisk od Alice dešifruje všem dostupným veřejným klíčem Alice. Dešifrování proběhne úspěšně pouze v případě, že byl otisk zašifrován soukromým klíčem Alice, který vlastní jedině ona. A pouze v případě úspěšného dešifrování otisku se bude dešifrovaný otisk zprávy shodovat s otiskem vypočteným Bobem a Alice bude považována za autentizovaného odesílatele přijaté zprávy.

TODO: najít vhodnější vysvětlení symetrického a asymetrického šifrování - funkčnost

Virtuální privátní sítě

VPN jsou mechanismem, umožňujícím organizacím budovat privátní sítě s použitím sdílené infrastruktury, např. veřejného internetu. Přitom je ale dosaženo stejné úrovně flexibility a bezpečnosti, jako při použití vlastní privátní infrastruktury. Princip spočívá v tunelování šifrovaných dat přes sdílenou infrastrukturu za použití autentizace mezi jednotlivými konci tunelů. Pakety jsou zabalené v nějakém protokolu. Obalujícím protokolem je v Internetu vždy IP, protokolem tunelovaným může být také IP, lze ale přenášet i jiné síťové protokoly (např. Novell IPX) nebo dokonce přímo rámce 2. vrstvy OSI RM.

Prostřednictvím VPN lze zajistit například připojení firemních notebooků kdekoliv na internetu do firemního intranetu (vnitřní firemní sítě). K propojení je třeba VPN server, který má přístup na internet i intranet (může sloužit pouze pro jednoho klienta, nebo sloužit jako hub a přijímat spojení od více klientů), a VPN klient, který se přes internet připojí k serveru a prostřednictvím něj pak do intranetu. VPN server pak plní v podstatě funkci síťové brány.

Použití VPN

- Virtuální propojení sítí
- Připojování vzdálených uživatelů

Implementace VPN na 3. vrstvě – IPSec

IPSec je architektura pro technickou realizaci tunelů. Poskytuje autentizace, integritu dat a šifrování. IPsec (IP security) je bezpečnostní rozšíření IP protokolu založené na autentizaci a šifrování každého IP datagramu. V architektuře OSI se jedná o zabezpečení již na síťové vrstvě, poskytuje proto transparentně bezpečnost jakémukoliv přenosu (kterékoliv síťové aplikaci).

SSL VPN

Secure Sockets Layer, SSL (doslova vrstva bezpečných socketů) je protokol, resp. vrstva vložená mezi vrstvu transportní (např. TCP/IP) a aplikační (např. HTTP), která poskytuje zabezpečení komunikace šifrováním a autentizaci komunikujících stran.

Protokol SSL se nejčastěji využívá pro bezpečnou komunikaci s internetovými servery pomocí HTTPS, což je zabezpečená verze protokolu HTTP. Po vytvoření SSL spojení (session) je komunikace mezi serverem a klientem šifrovaná a tedy zabezpečená.

Ustavení SSL spojení funguje na principu asymetrické šifry, kdy každá z komunikujících stran má dvojici šifrovacích klíčů - veřejný a soukromý. Veřejný klíč je možné zveřejnit, a pokud tímto klíčem kdokoliv zašifruje nějakou zprávu, je zajištěno, že ji bude moci rozšifrovat jen majitel použitého veřejného klíče svým soukromým klíčem.

SSL (Secure Sockets Layer) jako alternativa k IPSec, pracující ovšem na aplikační vrstvě, je pro budování VPN slabší z hlediska bezpečnosti (nezabezpečuje veškerou komunikaci, ale pouze některé aplikace - typu klient-server), ale je méně náročná na implementaci. Nepotřebuje nový klientský software, protože běží na standardních webových prohlížečích, a nevyžaduje tedy od uživatele žádnou instalaci. SSL ale nepodporuje všechny aplikace: hodí se pro zabezpečení komunikace webové, elektronické pošty nebo sdílení souborů. Naproti tomu se nehodí pro relačně orientované aplikace.

SSH

SSH neboli Secure Shell je klient/server protokol v síti TCP/IP, který umožňuje bezpečnou komunikaci mezi dvěma počítači pomocí transparentního šifrování přenášených dat. Pracuje na portu TCP/22. Pokrývá tři základní oblasti bezpečné komunikace: autentizaci obou účastníků komunikace, šifrování přenášených dat a integritu dat.

SSH je ve počítačové terminologii používán jako název přenosového (síťového) protokolu i jako název programu zprostředkovávající spojení. Data jsou přenášena mezi dvěma počítači přes nebezpečnou vnější síť vždy šifrovaně a volitelně s použitou kompresí.

Označení „Secure Shell“ je mírně zavádějící, protože nejde ve skutečnosti o shell ve smyslu interpret příkazů.

SSH program je dnes běžně používán při vzdálené práci a pro vzdálenou správu. Většinou se spojuje s SSH démonem (SSH daemon, sshd) pro navázání spojení. SSH démon rozhoduje podle svého nastavení, zda spojení přijme, jakou formu autentizace bude požadovat, případně na kterém portu naslouchá. Implementace SSH klientů i serverů (SSH démon) je dostupná na téměř jakékoli platformě.

TODO – VPN, lépe vysvětlit IPSec