IV Počítače a sítě

1. Architektura univerzálních procesorů. Principy urychlování činnosti procesorů.

Architektura mikroprocesorů

Architektura procesoru je náčrt struktury a funkčnosti procesoru. Architektura mikroprocesorů je charakterizována výčtem **registrů** a jejich funkcí, vnitřních a vnějších **sběrnic**, způsobem **adresování** a **instrukční souborem**.

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

Sběrnice je soustava vodičů pro přenos informací mezi více účastníky na principu jeden vysílá ostatní přijímají. Podle typu přenášené informace je dělíme na datové, adresové a řídící. V praxi však díky multiplexu může jít o jedny dráty.

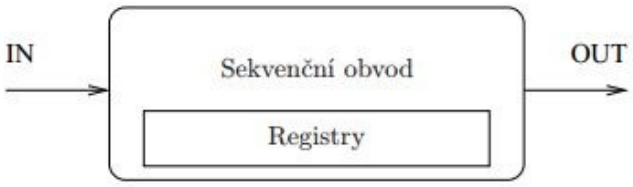
Procesory CISC a RISC

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

- 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 (sčítánía bitové posuvy), které mohou být ve výsledku vykonány rychleji, než hardwarově implementovaná složená varianta.
- Označení CISC bylo zavedeno jako protiklad až poté, co se prosadily procesory RISC, které mají instrukční sadu naopak maximálně redukovanou (pouze jednoduché operace, tj. žádné složené, jsou stejně dlouhé a jejich vykonání trvá stejnou dobu).
- Obvyklou chybou je domněnka, že procesory CISC mají více strojových instrukcí, než procesory RISC. Ve skutečnosti nejde o absolutní počet, ale o počet různých druhů operací, které procesor sám přímo umí vykonat na hardwarové úrovni (tj. již z výroby). Procesor CISC tak může například paradoxně obsahovat pouze jednu strojovou instrukci pro danou operaci (např. logická operace), zatímco procesor RISC může tuto operaci obsahovat jako několik strojových instrukcí, které stejnou operaci umí provést nad různými registry



Obrázek 1: Procesor (CISC) jako sekvenční obvod

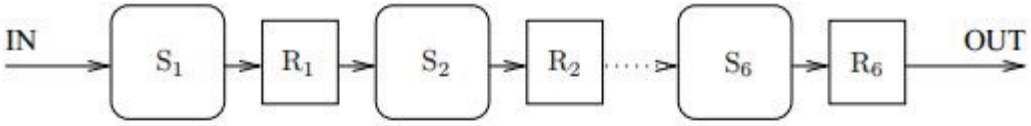
Krok		Význam
1.	**VI**	Výběr Instrukce
2.	**DE**	Dekódování
3.	**VA**	Výpočet Adresy
4.	**VO**	Výběr Operandu
5.	**PI**	Provedení Instrukce
6.	**UV**	Uložení Výsledku

	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

RISC

- Reduced Instruction Set Computer = počítač s redukovaným souborem instrukcí
- počet instrukcí a způsobů adresování je malý, ale zůstává úplný, aby bylo možno provést vše -> v tom se liší od CISC, což je procesor s velkým počtem instrukcí
- instrukce jsou vytvořeny pomocí obvodu -> jednodušší na výrobu než CISC
- širší sběrnice, rychlejší tok instrukcí a dat do procesoru
- instrukce jen nad registry
- navýšený počet registrů ->delší program
- instrukce mají jednotný formát – délku i obsah
- komunikace s pamětí pouze pomocí instrukcí LOAD/ STORE
- každý strojový cyklus znamená dokončení jedné instrukce
- používá se zřetězené zpracování instrukcí
- řešení problému s frontou instrukcí
- mikroprogramový řadič může být nahrazen rychlejším obvodovým
- přenášejí složitost technologického řešení do programu (překladače)
- ředtavitelé : ARM, MOTOROLA 6800, INTEL i960, MIPS R6000



Obrázek 2: Zřetězené zpracování (v procesoru RISC)

	T1	T2	T3	T4	T5	T6	T7	T8	T9	T10	T11	T12
VI	I1	I2	I3	I4	I5	I6	I7	...				
DE		I1	I2	I3	I4	I5	I6	I7				
VA			I1	I2	I3	I4	I5	I6	I7			
VO				I1	I2	I3	I4	I5	I6	I7		
PI					I1	I2	I3	I4	I5	I6	I7	
UV						I1	I2	I3	I4	I5	I6	I7

Tabulka 5: Zřetězené provádění instrukcí procesorem RISC

Urychlení činnosti procesoru:

- **paralelní zpracování** (násobné výpočetní jednotky)
- **zřetězené zpracování** (=pipeling)
- **přetaktování** (zvýšení činnosti procesoru)

Problémy zřetězeného zpracování

Zřetězené zpracování instrukcí (pipelining)

Na dosažení zřetězení je nutné rozdělit úlohu do **posloupnosti dílčích úloh**, z nichž každá může být vykonána samostatně, např. oddělit načítaní a ukládání dat z paměti od provádění výpočtu instrukce…a tyto části pak mohou běžet souběžně.

To znamená že musíme osamostatnit jednotlivé části sekvenčního obvodu tak, aby každému obvodu odpovídala jedna fáze zpracování inst. Všechny fáze musí být stejně časově náročné, jinak je rychlost degradována na nejpomalejší z nich. Fáze zpracování je rozdělena minimálně na 2 úseky:

- načtení a dekódování instrukce
- provedení instrukce a případné uložení výsledku

Zřetězení se stále vylepšuje a u novějších procesorů se již můžeme setkat stále s více řetězci rozpracovaných informací (více pipelines), dnes je standardem 5 pipelines.

Problém:

Největší problém spočívá v plnění zřetězené jednotky, hlavně při provádění podmíněných skoků, kdy během stejného počtu cyklů se vykoná více instrukcí.

U pipelingu se instrukce následující po skoku vyzvedává dřív než je skok dokončen. Primitivní implementace vyzvedává vždy následující instrukci, což vede k tomu že se vždy mýlí, pokud je skok nepodmíněný. Pozdější implementace mají jednotku předpovídání skoku (1 bit), která vždy správně předpoví nepodmíněný skok a s použitím cache se záznamem předchozího chování programu se pokusí předpovědět i cíl podmíněných skoků nebo skoků s adresou v registru nebo paměti.

V případě, že se predikce nepovede, bývá nutné vyprázdnit celou pipeline a začít vyzvedávat instrukce ze správné adresy, což znamená relativně velké zdržení. Souvisejícím problémem je přerušení.

Plnění fronty instrukcí

Pokud se dokončí skoková instrukce která odkazuje na jinou část kódu musejí být instrukce za ní zahozeny (problém plnění fronty intrukcí).

- u malého zřetězení neřešíme
- používání bublin na vyprázdnění pipeline, naplnění prázdnými instrukcemi
- **predikace skoku** - vyhrazen jeden bit předurčující, zda se skok provede či nikoliv

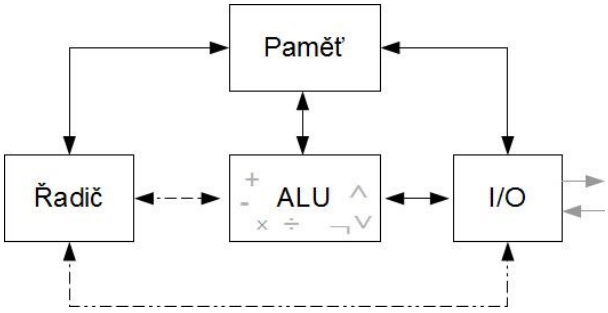
- statická – součást instrukce, řeší programátor
- dynamická

    - jednobitová – zaznamenává jestli se skok provedl či ne (1/ 0) - příště bude vědět
    - dvoubitová - metoda zpožděného skoku, v procesoru řeší se např. tabulkou s 4kB instr.

- ***superskalární architektura*** (zdvojení) - když nastane podmíněný skok, začnou se vykonávat instrukce obou variant, nepotřebná část se pak zahodí. Tento způsob, pak vyžaduje vyřešit ukládání výsledku.

Von Neumannovo schéma počítače

Von Neumannovo schéma počítače bylo navrženo v 40.letech Neumannem a v podstatě se dodnes příliš nezměnila. Je postavena na těchto pricipech:



- architektura je nezávislá na řešeném problému
- paměť rozdělená do buněk stejné velikosti
- program dán sekvencí instrukcí, která je změněny pouze skokem (podmíněným skokem)
- využita dvojkovou soustavou

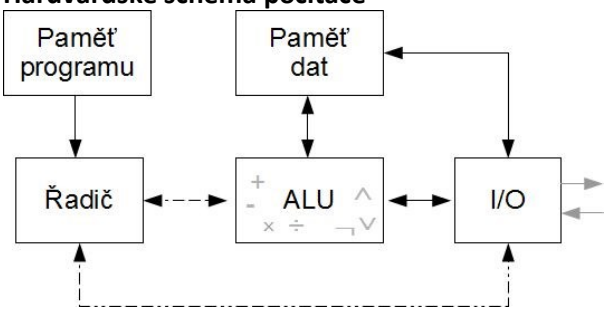
Jak je vidět na obrázku vpravo, skládá se Von Neummanová architektura z následujících částí:

- **Operační paměť** - slouží k uchování programu, dat a výsledků výpočtů. Jelikož přístup do hlavní paměti je časově náročný vznikla **zásobníková paměť** (skupina rychlých registrů) ke které se přistupuje sekvenčně jako na haldu.
- **ALU - Arithmetic-Logic Unit** - výpočetní jednotka, provádí aritmetické a logické operace.
- **Řadič** - řídicí jednotka řídí činnost všech částí počítače pomocí řídících signálů. Řadiči zpět jsou zasílané stavové hlášení.
- **I/O** - zařízení určená pro vstup a výstp

CPU (processor) = Řadič + ALU

Od dnešních počítačů se Von Neumannová architektura liší v množství použitých procesorů. Von Neumann počítá s jedním procesorem, zatímco **dnešní počítače** dokáží rozdělit výpočty mezi **více procesoru**. Podobně je to s množstvím spuštěných programů, dnešní počítače jsou schopny paralelně zpracovávat **více programů**, které navíc nemusí být celé, může se vykonat jen **část programu** a zbytek se zavede v případě potřeby.

Hardvardské schéma počítače



Hardvardské schéma počítače má na rozdíl od Von Neumannova **dvě oddělené paměti**. Jedna slouží k uložení programu a druhá pro proměnná data. Přičemž je vyloučeno, aby se paměti používaly jinak. Tato architektura dr používá např. pro jednoúčelné programové automaty či **kalkulačky**.

2. Základní vlastnosti monolitických počítačů a jejich typické integrované periférie. Možnosti použití.

Základní vlastnosti monolitických počítačů a jejich typické integrované periférie.

Jak ze základní architektury počítače víme, skládá se počítač ze tří částí:

- procesor (CPU)
- paměť
- periférie.

Pokud některá z těchto tří částí chybí, nejde o počítač. Pro řadu nenáročných aplikací se již přes 30 let vyrábějí malé počítače integrované v jediném pouzdře. Tvoří tedy jeden celek - monolit. Odtud tedy obecně používaný a zažitý název - monolitické počítače.

Pro monolitické počítače se převážné využívá harvardská architektura, protože přináší důležitou vlastnost: oddělení paměti pro data od paměti programu. To umožnuje snadno konstruovat počítač, který má paměť dat i programu vyrobenou jinou technologií a navíc dovoluje mít odlišnou velikost nejmenší adresovací jednotky. Např. data jsou obvykle uložena po bytech. A strojové instrukce procesoru mohou mít u každého procesoru jinou velikost, např. 12, 14, 16, 24 nebo 32 bitu a nejmenší adresní jednotka paměti programu se této velikosti přizpůsobuje.

Další dělení počítačů je obvykle podle použitého procesoru, tedy CISC a RISC. V dnešní době kvůli jednoduchosti nesou monolitické počítače převážné rysy architektury RISC, i když často ve velmi zjednodušené formě.

Paměti

- Pro data používáme většinou paměti energeticky závislé typu **RWM-RAM** (Read-Write Memory - Random Access Memory), tedy pamět s libovolným přístupem určenou pro čtení i zápis. Tyto paměti jsou vyráběny jako statické (uchování paměti po celou dobu napájení), jejich paměťové buňky jsou realizovány jako klopné obvody.
- Pro program se používají paměti, které si svůj obsah zachovají i po odpojení napájení, tedy jde o paměti typu ROM určeny především ke čtení. Nejčastěji paměti **EPROM**, EEPROM a **Flash**. Nesmíme také zapomenout na výrobky s pamětí **PROM**.

Organizace paměti

- **Střadačové (pracovní) registry** - ve struktuře procesoru jsou obvykle 1-8-16 základních pracovních registrů, jsou nejpoužívanější. Ukládají se do nich aktuálně zpracovávaná data a jsou nejčastějším operandem strojových

instrukcí (to na co se instrukce v závorkách odkazují). A také se do nich nejčastěji ukládají výsledky operací. Nejsou určeny pro dlouhodobé ukládání dat.

- **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 (z těch předešlých nejméně používaný). Instrukční soubor obvykle nedovoluje s obsahem této paměti přímo manipulovat, kromě instrukcí přesunových. Těmi se data přesunou např. do pracovního registru. Některé procesory dovolují, aby data z této paměti byla použita jako druhý operand strojové instrukce, výsledek ale nelze zpět do této paměti uložit přímo. Konstrukčně je řešeno pamětí EEPROM.

Zdroje synchronizace

- krystal (křemenný výbrus) – jsou drahé ale přesné
- keramický rezonátor
- obvod RC – snadno integrovatelný
- obvod LC – méně časté

Ochrana proti rušení

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. Tam se také řadí **BROWN-OUT** ochrana proti podpětí (-> reset).

Typické periferie

Periferie - obvody, které zajištují komunikaci mikropočítace s okolím

- **Vstupní a výstupní brány -** Nejjednodušší a nejcasteji 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. Na vstupu je Schmittův klopný obvod. U mnoha mikropočítačů jsou brány implementovány tak, že s nimi instrukční soubor může pracovat jako s množinou vývodu, nebo jako s jednotlivými bity.
- **Čítace a časovače** - do skupiny nejpoužívanejších periférií mikropočítače určite patří čítače a časovače. Časovač se od čítače příliš neliší. Není ale inkrementován vnejším signálem, ale přímo vnitřním hodinovým signálem používaným pro řízení samotného mikropočítače. Lze tak podle přesnosti zdroje hodinového signálu zajistit řízení událostí a chování v reálném čase. Pri přetečení časovače se i zde může automaticky předávat signál do přerušovacího podsystému mikropočítače.
- **Sériové linky** - Sériový přenos dat je v praxi stále více používán. Dovoluje efektivním způsobem prenášet data na relativně velké vzdálenosti pri použití minimálního počtu vodičů. Hlavní nevýhodou je však nižší přenosová rychlost, a to že se data musí kódovat a dekódovat.
    - **USART** (RS232) +/-12V jet transformována na TTL /RS422/RS485
    - **I2C** (Philips) komunikace mezi integrovanými obvody (přenos dat uvnitř elektronického zařízení)
    - **SPI**

- **A/D a D/A převodníky -** Fyzikální veliciny, 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. Existuje několik základních typu těchto převodníků:

- A/D – velký počet součástek, malá rozlišovací schopnost, velmi rychlá
- D/A prevodníky – PWM (šířková modulace pulzu), vyroben pomocí odporů, jednoduchá konstrukce, rychlá odezva, velká rozlišovací schopnost, vyrobit sadu odporů vyžaduje velkou přesnost

3. Struktura OS a jeho návaznost na technické vybavení počítače.

Struktura OS

Co vše provádí operační systém

- Organizuje přístup 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 programů a mylné požadavky uživatelů tak, aby tyto chyby nezpůsobily zásadní destrukci systému nebo poškození dat.
- Spravuje komunikaci s periferiemi. Definuje nastavení klávesnice, citlivost myši a dalších zařízení.
- Eviduje využívání systémových zdrojů apod.

Operační systém je zpravidla tvořen tzv. jádrem (kernel), ovladači V/V zařízení (driver), příkazovým procesorem (shell) a podpůrnými systémovými programy.

- **Jádro** - po zavedení do paměti řídí činnost počítače, poskytuje procesům služby a řeší správu prostředků a správu procesů.
- **Ovladač** - zvláštní (pod)program pro ovládání konkrétního zařízení standardním způsobem. Použití strategie s ovladači umožňuje snadnou konfigurovatelnost technického vybavení.
- **Příkazový procesor** - program, který umožňuje uživatelům zadávat příkazy ve speciálním, obvykle jednoduchém jazyce.
- **Podpůrné programy** - do této kategorie jsou mnohdy zahrnovány i překladače (jazyk C v OS UNIX) a sestavující programy. Stojí na stejném místě jako aplikační programy.

Jádro OS

Jádro se zpravidla dělí na dvě podstatné části:

1. ***Správa procesů*** - správa procesů (prakticky není u jednoduchých OS) řeší problematiku aktivování a deaktivování procesů podle jejich priority resp. požadavků na prostředky.
2. ***Správa prostředků*** - zajišťuje činnost V/V zařízení, přiděluje paměť, případně procesory. Velmi důležitou částí správy prostředků 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.

Generické komponenty OS

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

Správa procesů

Proces – činnost řízená programem, provedení programu, „process“ nebo „task“

- Proces potřebuje pro svou realizaci jisté zdroje (CPU, paměť, I/O zařízení, …)
- Z hlediska správy procesů je OS zodpovědný za:
    - Vytváření a rušení procesů
    - Potlačení a obnovení procesů
    - Synchronizace procesů a jejich vzájemná komunikace
- Variantou procesu je tzv. Vlákno – jednotka plánování činností definovaná v programu. Vlákna používají zdroje přidělené procesu.

Správa procesoru

- OS vybírá procesy běžící na dostupných procesorech.
- Dále podle typu OS zde patří i plánování vláken.

Správa vnitřní paměti

- Vnitřní pamětí se myslí primární, operační paměť, neboli fyzický adresový prostor (pole samostatně adresovatelných slov nebo bytů), repositář pohotově (rychle) dostupných dat sdílený CPU a I/O zařízeními.
- Vnitřní paměť energeticky závislá, ztrácející se po výpadku energie.
- OS je z hlediska správy vnitřní paměti odpovědný za:
    - Vedení přehledu, který proces v daném okamžiku využívá kterou část paměti
    - Rozhodování, kterému procesu po uvolnění paměti tuto paměť poskytnout
    - Přidělování a uvolňování paměti
- Programátor vidí logický adresový prostor – ten je vymezen formou adresy v instrukci. OS zavádí programy a data do tohoto prostoru podle potřeby. Transformaci logické adresy na fyzickou se zpravidla provádí až při provádění instrukce. Struktura logického adresového prostoru může být lineární (pole) nebo dvojdimensionální (kolekce samostatných lineárních segmentů proměnné délky).

Správa I/O systémů

Poskytuje repositář vyrovnávacích pamětí a univerzální rozhraní ovladačů (driverů) I/O zařízení, a pak také samotné ovladače (drivery).

Správa vnější paměti

Správa energeticky nezávislé, sekundární paměti s dostatečnou kapacitou. Jako sekundární paměť obvykle slouží disky. OS je zodpovědný za:

- Správu volné paměti
- Přidělování paměti souborům
- Plánování činnosti disku

Správa souborů

Soubor – identifikovatelná kolekce souvisejících dat definovaná svým tvůrcem. Obvykle reprezentuje i programy i data.

OS je z hlediska správy souborů zodpovědný za:

- Vytváření a rušení souborů
- Vytváření a rušení adresářů (katalogů)
- Podporu primitivních operací pro manipulaci se soubory a adresáři
- Zobrazování souborů do sekundární paměti
- Archivování souborů na energeticky nezávislá média
- Networking, distribuované systémy

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.

Systém ochran musí:

- Rozlišovat mezi autorizovaným a neautorizovaným použitím
- Poskytnout prostředky pro své prosazení

Interpret příkazů

Většina zadání je předávána operačnímu systému řídícími příkazy, které zadávají požadavky na:

- Správu a vytváření procesů
- Ovládání I/O zařízení
- Správu sekundární paměti
- Práci v síti

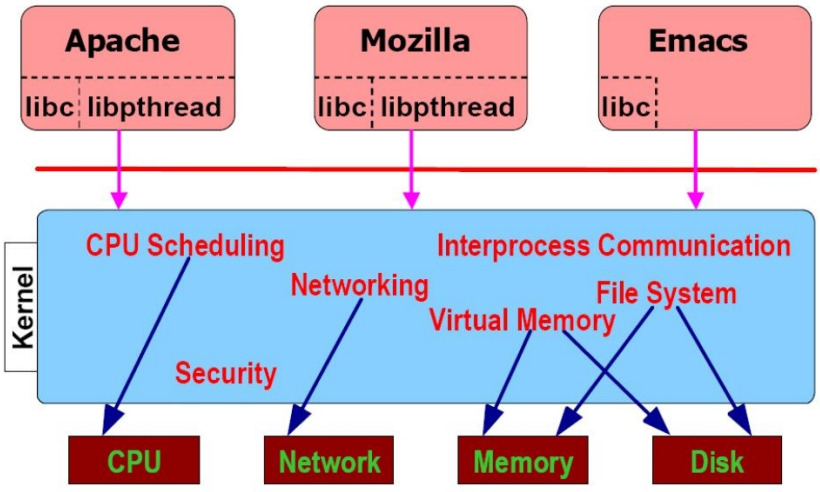
Program, který čte a interpretuje příkazy je označován v různých OS různými názvy – command line, interpreter, shell, …

Struktura OS podle jádra

- Monolitická jádra (Monolithic kernel)
- Otevřené systémy (Open systems)
- Microkernel

Monolitický OS

OS je na jednom místě (pod „červenou čárou“). Aplikace používá dobře definované rozhraní systémových volání pro komunikaci s jádrem.



Příklady OS – Unix, Windows NT / XP, Linux, BSD, … Monolitické jádro je specifické pro komerční systémy.

Výhody:

- Dobrý výkon
- Dobře pochopená koncepce
- Jednoduché pro vývojáře jádra
- Vysoká úroveň ochrany mezi aplikacemi

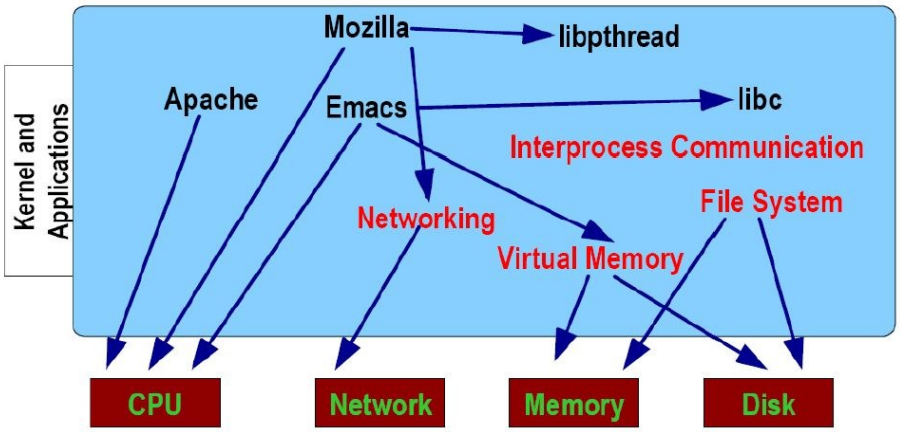
Nevýhody:

- Žádná ochrana mezi komponentami jádra
- Není jednoduše a bezpečně rozšiřitelné jádro
- Celková struktura je ve výsledku komplikovaná, neexistují hranice mezi moduly jádra

Otevřené Systémy

Aplikace, knihovny i jádro jsou všechny v jednom adresovém prostoru.

Jestli se to zdá být jako šílená myšlenka, zde jsou příklady – MS-DOS, Mac OS 9 a starší, Windows ME, 98, 95, 3.1, Palm OS.



Tato koncepce bývala velmi běžná.

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

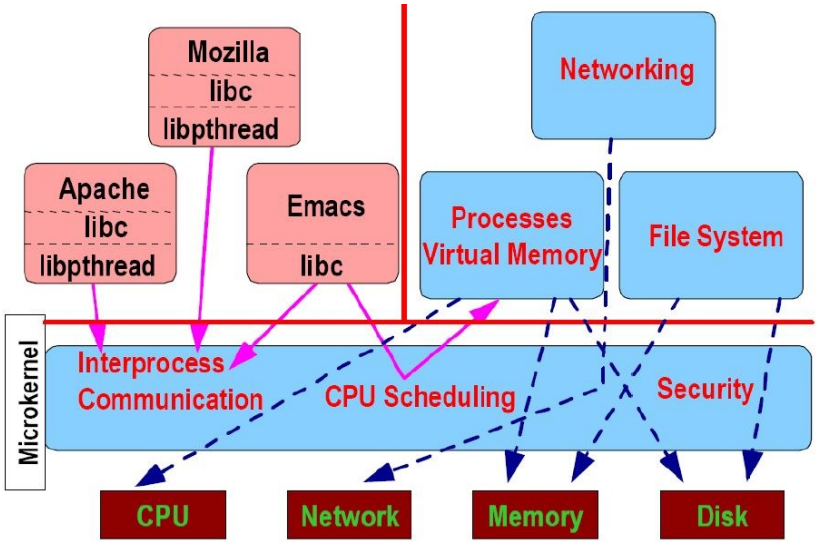
Microkernel OS

Filosofie návrhu – chráněné jádro obsahuje pouze minimální (malou, čistou, logickou) množinu abstrakcí:

- Procesy a vlákna
- Virtuální paměť
- Komunikace mezi procesy

Všechno ostatní jsou server-procesy běžící na uživatelské úrovni.

Příklady OS – Mach, Chorus, QNX, GNU Hurd



Zkušenosti s tímto návrhem jsou smíšené.

Výhody:

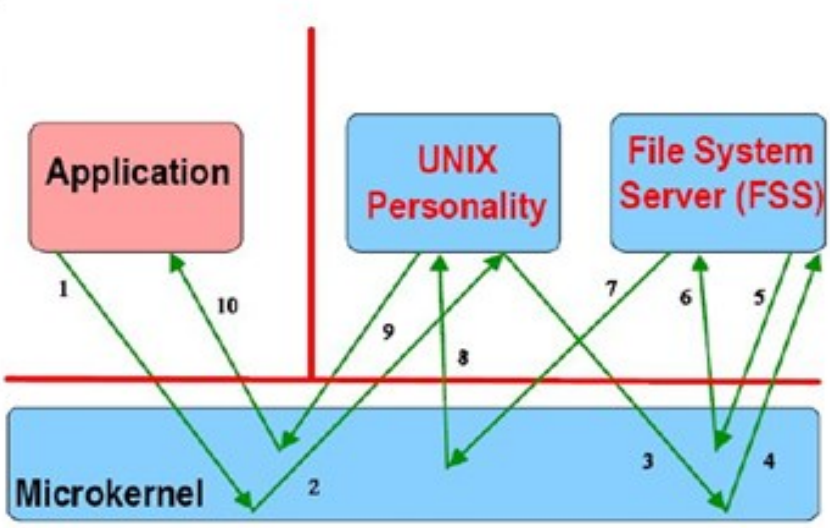
- Přidáním server-procesu se rozšíří funkcionalita OS
- Jádro nespecifikuje prostředí OS
- Servery na uživatelské úrovni se nemusí zabývat hardwarem
- Silná ochrana OS i sám proti sobě (části OS jsou oddělené servery)
- Jednoduché rozšíření na distribuovaný nebo multiprocesorový systém

Nevýhody

- Výkon (Systémová volání viz. dále)
- Špatná minulost

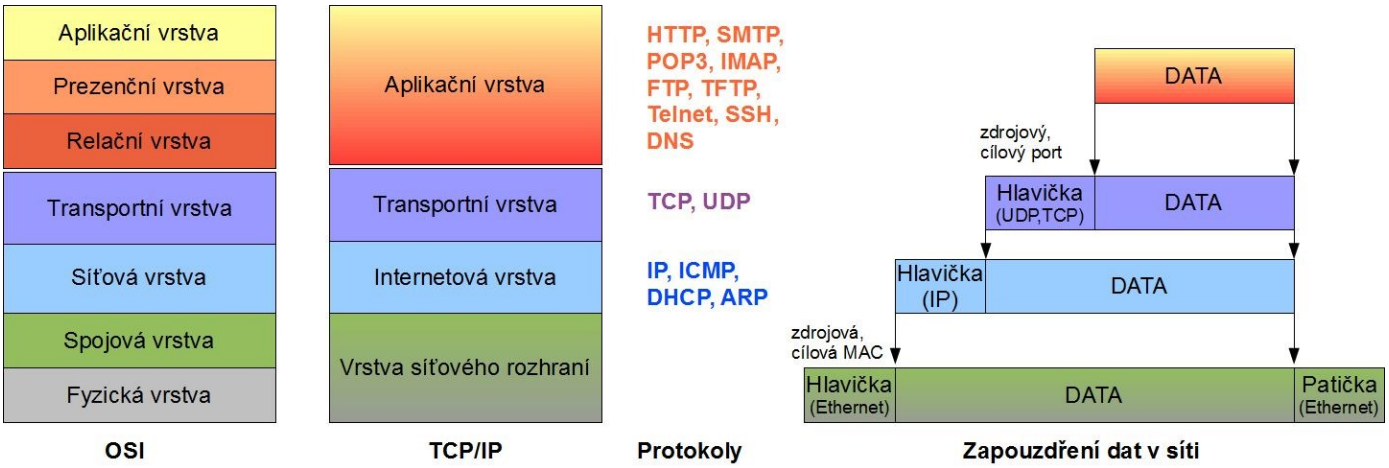
Systémová volání – příklad

1. Application calls read(), traps to microkernel
2. microkernel sends message to Unix Personality requesting read
3. Unix personality sends message to File System Server (FSS) asking for data
4. FSS receives message and begins processing
5. FSS sends message to microkernel asking for disk blocks
6. Microkernel sends data back to FSS
7. FSS sends message to UNIX Personality with results
8. Unix Personality receives message with data
9. Unix Personality sends data to Application
10. Application receives data



4. Protokolová rodina TCP/IP.

TCP/IP je rodina protokolů pro komunikaci v síti a tedy i Internetu. Následující obrázek ukazuje vztah mezi OSI a TCP/IP a ukázku zapouzdření dat v jednotlivých vrstvách



Protokoly aplikační vrstvy

Je tvořena množinou protokolů spolupracujících s jednotlivými aplikačními programy. Jejich funkci se pokusím vysvětlit na jednoduchém příkladu:

Při prohlížení webových stránek používáme prohlížeč. Prohlížeče spolupracují s internetovými servery, na nichž jsou webové stránky uloženy a uživatelům nabízeny. Vidíme, že při brouzdání Internetem spolupracují různé programy různých výrobců. Jejich spolupráce je zajištěna aplikačním protokolem – vlastně soustavou norem, které musí tyto programy respektovat. (Při prohlížení www stránek jím je protokol http.)

- **HTTP** (HyperText Transfer Protokol) - internetový protokol pro výměnu HTML dokumentů.
- **SMTP** (Simple Mail Transfer Protokol) - internetový protokol pro přenos e-mailů. Zajišťuje přímé spojení mezi odesílatelem a adresátem (poštovní schránkou).
- **IMAP, POP3** - jsou dva protokoly které umožňují výběr pošty ze schránky. IMAP umožňuje různě manipulovat s obsahem schránky, při zobrazení složky stahuje pouze hlavičky a jejich obsah až v případě, že si je uživatel chce přečíst. POP3 pouze stahuje zprávy ze vzdáleného serveru na lokální umístění (změny jako mazání pošty se nepromítají na serveru).
- **FTP** (File Transfer Protokol) - protokol pro přenos souboru mezi dvěma počítači, na kterých můžou běžet rozdílne operační systémy. Pracuje na principu klient-server na portech TCP/20 a TCP/21 (20 k přenosu dat a 21 k řízení)
- **TFTP** (Trivial FTP) - zjednodušený FTP, obsahuje jen jen základní funkce FTP. Používá se v případech kdy FTP je kvůli své složitosti nevhodný (bootovaní bezdiskových počítačů ze sítě).
- **Telnet** - klient-server protokol, který umožňuje připojit se z klientského terminálu na vzdálenou stanici. Po navázání spojení může klient komunikovat se vzdáleným počítačem, jako by seděl přímo u něj.
- **SSH** (Secure Shell) - protokol pro zabezpečenou (šifrovanou) komunikaci. Byl navržen jako náhrada za Telnet a další nezabezpečené přístupy.
- **DNS protokol** (Domain Network System) - zajišťuje převod doménových jmen na IP adresy.

Protokoly transportní vrstvy

Je jakýmsi jádrem celé soustavy TCP/IP, tvořeným pouze dvěma protokoly: TCP a UDP.

- **TCP** (Transmision Control Protocol) se stará o spolehlivé doručování dat ve správném pořadí. Kontrolním součtem ověřuje, jestli data nebyly po cestě poškozená a TCP modul na straně příjemce pak zasílá zpět zprávy o úspěšně přijatých paketech.
- **UDP** (User Datagram Protokol) - stejně jako TCP se stará o přenos dat, ale není spolehlivý. Nezaručuje správné pořadí paketů, přidá pouze kontrolní součtet. Používá se u přenosů, kde nám nevadí, že se něco stratí (streamování videa).

Protokoly internetové (síťové) vrstvy

Tato skupina protokolů je dnes určitě nejrozšířenější. Původně byla navržena pro síť, z níž se vyvinul Internet. Dnes je rodina protokolů TCP/IP používána v sítích Novellu i Microsoftu, kde se stala standardem a své předchůdce zcela vytlačila.

Z funkčního hlediska můžeme TCP/IP rozdělit na tři vrstvy (reprezentované samostatnými protokoly):

IP (Internet Protokol) - základní protokol pro přenos dat po sítí. Sám nezaručuje nic. Podle IP adres jen směruje paket k cíli.

ICMP (Internet Control Message Protocol) - slouží pro odesílání chybových zpráv např. že služba není dostupná.

ARP (Adress Resolution Protocol) - převádí IP adresy na MAC adresy

DHCP (Dynamic Host Configuration Protocol) - požívá se pro automatickou konfiguraci počítačů připojených do sítě. Přiděluje například IP adresy.

Ethernet

Ethernet je v informatice souhrnný název pro v současné době nejrozšířenější technologie pro budování počítačových sítí typu LAN (tj. domácí nebo firemní sítě). Ethernet se stal de facto standardem pro svoji jednoduchost a nízkou cenu a vytlačil z trhu ostatní alternativní technologie (např. ARCNET,ATM, FDDI). V současné době je ethernetové rozhraní s konektorem RJ-45 pro kroucenou dvojlinku standardním síťovým rozhraním prakticky všech notebooků, netbooků i základních desek běžných stolních počítačů.

5. Metody sdíleného přístupu ke společnému kanálu.

Sdílení přístup ke kanálu

Metody sdílení přenosového kanálu se dělí na:

- **deterministické** (bezkolizní) - řídí se algoritmem, který přesně definuje kdo, kdy bude vysílat a ke kolizím nedochází
- **nedeterministické** (kolizní) - algoritmus je založen dost na náhodě (náhodné časové prodlevy) a musí řešit kolize, tedy situace kdy chce naráz na kanálu vysílat více stanic

Nedeterministické metody sdíleného přístupu

Není dán jednoznačný algoritmus na rozdělení přístupu jednotlivých stanic ke kanálu, proto se musí řešit kolize.

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

Aloha

Metoda vhodná pro kanál s řídkým provozem. Vznikla pro radiovou síť na Havajských ostrovech.

- **Prostá aloha** - netestuje se obsazenost kanálu a rovnou se vysílá. Nepřijde-li do časového limitu potvrzení, pokus se opakuje. *Kolizní slot* = dvojnásobku doby vysílaní rámce (když začnu vysílat na konci vysílaného rámce druhé stanice)
- **Taktovaná aloha** - vysílací čas rozdělen na takty v kterých se smí začít vysílat. Díky tomu nedochází k tomu, aby jedna stanice začala vysílat ve chvíli, kdy skoro skončila druhá a *kolizní slot* je poloviční.

- **Řízená aloha** - řídí intenzitu opakování podle zatížení sítě. Čím zatíženější je síť, tím je interval po kterém je možné zopakovat pokus delší.

CSMA

Metody náhodného přístupu, které mají informaci o obsazenosti kanálu.

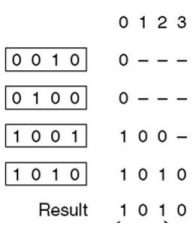
- **Nalehájící CSMA** - testuje stav kanálu. Pokud je obsazený, počká s vysíláním na jeho uvolnění. Je zde riziko kolize čekajících stanic. Takové kolize se pak řeší náhodnou dobou čekání pro zopakování pokusu.
- **Nenaléhající CSMA** - při detekci obsazeného kanálu čeká náhodnou dobu na další pokus. Čekací doba se často volí jako k-násobek doby průchodu signálu sdíleným médiem.
- **P-nalehací CSMA** - naléhá na vysílání s pravděpodobnosti *p*, s pravděpodobnosti *1-p* počká na další timeslot. Pro *p=1* jde o naléhací CSMA
- **CSMA/CD** - CSMA s detekcí kolize. Při detekci kolize je zastaveno vysílání a čeká se náhodnou dobu pro další pokus. Před vysíláním musím být na mediu klid po dobu jednoho kolizního slotu.
  CSMA/CD v Eternetu vysílá při detekci kolize kolizní signál 'jam', aby kolizi rozpoznaly všechny kolidující stanice. Prodlevu po kolizi určí algoritmus Binary exponential backoff.

Deterministické metody pro sdílení přenosového media

- **CSMA/CA** - bezkolizní
  - zařízení si od řídícího prvku vyžádá povolení vysílat
  - pokud je mu uděleno, začne vysílat a ostatní zařízení čekají
  - používá se u WiFi

Deterministické metody se dělí na:

- **Centralizované řízení** - jedna ze stanic je master a přiděluje ostatním právo vysílat. Je to sice efektivní ale musí se obětovat část kanálu pro komunikaci s mastrem.
  - **Přidělování na výzvu** - stanice smí vysílat jen, když je k tomu vyzvána masterem
    - výzva je cyklicky opakována a stanice buď odešle něco, nebo mlčí
    - binární vyhledávání - master vyhledává stanici připravenou vysílat (vhodné pro velké množství stanic a řídkému vysílání)
  - **Přidělování na žádost** - master zpracovává žádosti jednotlivých stanic a přiděluje jim kanál. Pro žádosti je často vyhrazen speciální kanál.
- **Distribuované řízení** - nezávislé na řídící stanici.
  - **Rezervační rámec** - master vysílá rezervační rámec, který má tolik slotu kolik je stanic a stanice si do svého slotu může umístit požadavek na vysílání. Datové sloty pak následují za rezervačním rámcem.
    Režie rezervačního rámce = N
  - **Binární vyhledávání** - stanice postupně, které chtějí vysílat posílají bity své adresy. Jakmile odešlou 0 a přečtou 1, chce vysílat někdo z vyšší prioritou a tak se odmlčí (viz. obr.) Stanice, která úspěšně odešle celou svou adresu, může vyslat jeden rámec.
    Režie binárního vyhledávání = log2N



  - **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.
    Problém je při přidávání stanic za provozu.

6. Problémy směrování v počítačových sítích. Adresování v IP, překlad adres (NAT).

Směrování v sítích

Pojmem **směrování (routing)** je označováno hledání cest v počítačových sítích. Jeho úkolem je dopravit datový paket určenému adresátovi, pokud možno co nejefektivnější cestou. Směrování je základním úkolem síťové vrstvy (L3) referenčního modelu OSI. Prakticky jej zajišťují routery (směrovače) v rámci WAN sítě (Internet).

Problém se smyčkami v síti

Switch je zmatený, když mu chodí zprávy z jedné adresy na různé porty. K čemuž dochází v případě, že síť obsahuje cykly.

Řešení: **Spanning Tree**
Graf sítě, ale bez cyklů, na základě kterého switch nepoužívá některé porty. V případě výpadku nějaké části si graf aktualizuje.

LAN sítě - jsou lokální sítě, tedy počítače propojené HUBy, nebo SWITCHi (může být i routery)

> **HUB** žádné směrování neřeší, to co mu příjde z jednoho portu pošle na všechny ostatní.
>
> **SWITCH** už je inteligentnější a eviduje si v tabulce kam má jakou adresu směrovat. Toto směrování probíhá na 2.vrstvě OSI modelu. Pokud však posílám data do světa, přichází na řadu router.

WAN sítě - velké sítě komunikující na velké vzdálenosti (Internet) jsou propojeny routry.

> **ROUTER**- zařízení k propojení sítí. Pracuje na 3. vrstvě OSI modelu a využívá protokol IP. Router si vede a aktualizuje **routrovací tabulku (směrovací tabulku)**, které obsahují nejlepší cesty k cílům.

Směrovací tabulka - Základní datovou strukturou, podle které se rozhoduje, co udělat s kterým paketem. Formát: *Cílová adresa/maska, Výstupní rozhraní, Brána (next hop), Metrika.* Použije se cesta, která se shoduje s adresou cíle v paketu na co největší počet míst.

Výchozí cesta – typická pro sítě připojené jediným rozhraním k hierarchicky vyšší síti. Pro ty pakety, pro které neexistuje položka ve směrovací tabulce, bude použita výchozí cesta. Snižuje počet záznamů v tabulce.

Konvergence - je proces a čas potřebný pro konverzi směrovacího protokolu. Dosažena ve chvíli, kdy všechny routry mají kompletní aktuální informaci o topologii.

Metrika - Každý směrovací protokol potřebuje kritérium, podle kterého posoudí, která z více možných cest do cílové sítě je nejvýhodnější. Toto kritérium se označuje jako metrika. Různé metriky více (IGRP-bandwidth+delay) či méně (RIP-počet přeskoků v síti) odrážejí skutečné vlastnosti linek.

Pro směrování se používají 2 základní algoritmy: ***statické*** a ***dynamické***.

Problém nejkratší cesty: Problém nalezení nejkratší cesty pro pakety.

Statické (neadaptivní) směrování

Záznamy v routovacích tabulkách nejsou za běhu měněny. Změnit je může jen správce počítače.

Dynamické (adaptivní) směrování

Tabulky jsou měněny a aktualizovány za provozu počítače a přizpůsobují se změnám v síti. Dynamické směrování dělíme na:

- **Centralizované** - při centralizovaném směrování routery posílají všechny informace o stavu sítě do řídícího centra, které je vyhodnotí, vypočítá routovací tabulky a rozesílá je zpátky routerům.
- **Distribuované** - každý router zná cesty ke svým sousedům (vzdálenost, funkčnost, přenosovou rychlost) a vyměňuje si s nimi i informace o směrování. Tak se šíří informace síti ze souseda na souseda.
- **Izolované** - routovací tabulky si vypočítává sám na základě procházejících paketů, nebo používá jiné metody, které s tabulkou nepracují.
    - **Záplavové směrování** - do paketu umístí počítadlo přeskoků a vyšle jej na všechny linky kromě té, z které přišel. Po dosažení limitu na počítadle se paket sám zruší a dál se nešíří.
    - **Backward learning** - do paketu se vkládá počítadlo a identifikace zdrojové sítě. Router z příchozích paketů zjistí jak daleko jsou sítě z kterých paket přichází.
    - **Horky brambor** - router se chce co nejrychleji zbavit paketu, tak jej zařadí do nejkratší výstupní fronty.
- **Hierarchické** směrování se používá v hierarchicky členěných sítí, které jsou rozděleny na autonomní oblasti. Směrovače těchto autonomních celků znají jen topologií svého celku.

- **Distance Vector** – vektory vzdálenosti – jednoduchá implementace, historicky starší
- **Link State** – stavy spojů – složitější, ale rychlejší konvergence

Distance vector

Směrovače neznají topologii sítě, pouze rozhraní, přes která mají posílat pakety do jednotlivých sítí a vektory vzdálenosti k těmto sítím. Na začátku obsahuje směrovací tabulka pouze přímo připojené sítě, tak jak byly nakonfigurovány administrátorem. Celá směrovací tabulka je periodicky zasílaná sousedům. Z příchozích směrovacích tabulek (vzdáleností sousedů od jednotlivých sítí) a výběrem nejlepší cesty si směrovač postupně upravuje svou směrovací tabulku. Pokud mi nějaké cesta nebyla dlouho inzerována, odstraním ji z tabulky.

Metrikou je počet přeskoků (hop count) do cíle. Nezohledňuje parametry spojů. Pomalá konvergence při změnách topologie (kvůli periodě). Zátěž od broadcastu směrovacích tabulek.

S každým záznamem ve směrovací tabulce je uložen čítač, který určuje jeho stáří (typicky 3xupdate_perioda). Po vypršení je cesta rozesílaná jako nedosažitelná (route poisoning).

Link state

Směrování na základě znalosti „stavu“ jednotlivých linek (funkčnost, cena). Směrovače znají topologie celé sítě a ceny těchto linek. Informace udržují v topologické databázi (všichni ji mají stejnou). Každý směrovač počítá strom nejkratších cest ke všem směrovačům (a k nim připojeným sítím) pomocí Dijkstrova algoritmu.

Neustálé sledování stavu linek zajišťuje komunikace mezi směrovači pomocí LSA Hello zpráv. Při změně okamžitě šíří informaci o stavu svého okolí všem ostatním. Spotřebovává více pásma (hlavně na počátku zasílá množství LSP) a zdrojů na routeru.

Topologická databáze obsahuje záznamy ve tvaru: ID směrovače, seznam přilehlých linek k sousedním směrovačům (u každé je ID souseda), seznam koncových sítí připojených ke směrovači. U každé linky je i její cena. Z těchto záznamů lze zkonstruovat graf topologie sítě.

Pro zlepšení vlastností se rozděluje na menší oblasti, hraniční routery posílají sumární cesty, využívá multicast, číslování LSA.

Adresování IP

Adresu sítě přiděluje oblastní správce (pro Evropu RIPE).

Adresu je 32bitová (8b.8b.8b.8b) a skládá se z adresy sítě + adresy uzlu v rámci sítě. Kolik bitů tvoří adresu sítě se dozvíme z masky (255.0.0.0 znamená že prvních 8b je adresa sítě). Soukromé sítě maji vyhrazené rozsahy adres (192.168.0.0 - 192.168.255.0, aj.) a nesmí být připojené na Internet přímo ale přes router, který přes proxy adresy přeloží.

IP adresy se dělí na:

- unicast - adresa jednoho konkrétního počítače
- multicast - adresa pro více počítačů najednou
- broadcast - adresa na všechny počítače. Šíří se jen v rámci segmentu počítače, dál není propuštěna (255.255.255.255)
- loopback - zpětnovazební adresa, pošle paket zpátky na vlastního počítač (127.0.0.1)

NAT (Network Address Translation, překlad síťových 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. NAT může být softwarového typu (Nat32, Kerio Winroute firewall), nebo hardwarového typu (router s implementací NAT). Překlady buď staticky (ručně) nebo dynamicky (Adresy se vybírají z rezervoáru adres = pool).

Vlastní Komunikace: Klient odešle požadavek na komunikaci, 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.

Probíhá na routrech, které k tomu využívá překladové tabulky.

- **Dynamický NAT** - používá se v případě kdy máme více počítačů než přidělených veřejných adres a chceme, aby měly všechny stroje přístup k veřejné síti. Pak veřejné adresy plavou v poolu a jsou přidělovaný dle potřeby.
- **Statický NAT** - mapuje konkrétní adresu vnitřní sítě na konkrétní adresu vnější sítě.

7. Bezpečnost počítačových sítí s TCP/IP: útoky, paketové filtry, stavový firewall. Šifrování a autentizace, virtuální privátní sítě.

Útoky

Denial of Service (DOS)

- cíl útoku je vyřazení služby z činnosti - bývá konstruován tak, že jeho cílem je v jednom čase útok na jednu konkrétní službu
- dochází k přehlcení požadavky a pádu nebo minimálně jeho nefunkčnosti
- někdy jako součást jiného útoku či zahlazení stop

Všechny typy se vyznačují několika společnými charakteristikami:

- Zaplavení provozu na síti náhodnými daty, které zabraňují protékání skutečných dat.
- Zabránění nebo přerušení konkrétnímu uživateli v přístupu ke službě.
- Narušení konfiguračního nastavení.
- Extrémnímu zatížení CPU cílového serveru.
- Vsunutím chybových hlášení do sekvence instrukcí, které můžou vést k pádu systému.
- Pád samotného operačního systému.

DoS pomocí chyb v implementaci IP

- **PingOfDeath** – odeslání příliš velkého paketu pomocí ping, nekontrolující příjemce se zhroutil
- **Teardrops** – využívá chyby při skládání fragmentovaných paketů (posílá nekorektní fragmenty)

DoS pomocí nedokonalostí TCP/IP

- **SYN flooding**
  - útočník zahájí navázání TCP spojení (pošle paket SYN)
  - cíl potvrdí (SYN ACK) a alokuje pro otevírané spojení zdroje
  - útočník ale nedokončí navázání spojení, místo toho zahajuje otevírání dalších a dalších spojení
  - cíl postupně vyčerpá své zdroje a přestane přijímat žádosti o spojení od regulérních klientů
  - řešení: zkrátit dobu čekání na potvrzení navázaného spojení od klienta, alokovat pro ně zdroje až po potvrzení
- **Land attack** – varianta SYN útoku, v žádosti o spojení je jako adresát i odesilatel uveden cílový stroj, ten se zahltí zasíláním potvrzení sám sobě
- **Smurf** – zahlcení cíle ICMP pakety (ping), jejich zpracování mívá někdy přednost před běžným provozem; útočník pošle žádost o ping všem (broadcast) a jako odesilatele uvede cíl útoku
- **DNS útok** – podobný předchozímu, jen místo ICMP používá DNS dotazy a odpovědi

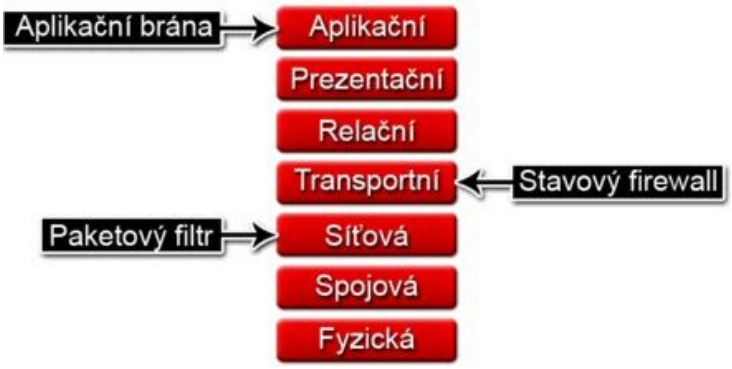
DDoS – Distributed Denial of Service

- DoS útok vedený souběžně z mnoha stanic
- na nezabezpečené počítače je distribuován útočný program (označován jako zombie), např. virem
- v určitý čas útočník vzbudí zombie a pošle je současně na cíl
- mnoho různých variant, zejména v přístupu k synchronizaci zombie
- obtížně se blokuje – zdrojů je příliš mnoho

Stavový firewall

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. Tato pravidla historicky vždy zahrnovala identifikaci zdroje a cíle dat (zdrojovou a cílovou IP adresu) a zdrojový a cílový port, což je však pro dnešní firewally už poměrně nedostatečné – modernější firewally se opírají přinejmenším o informace o stavu spojení, znalost kontrolovaných protokolů a případně prvky IDS. Firewally se během svého vývoje řadily zhruba do následujících kategorií:



- Paketové filtry
- Aplikační brány
- Stavové paketové filtry
- Stavové paketové filtry s kontrolou známých protokolů a popř. kombinované s IDS

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. Reflexivní ACL - Automaticky propouští vstupní provoz, který odpovídá povolenému provozu výstupnímu

Výhodou tohoto řešení je vysoká rychlost zpracování, proto se ještě i dnes používají na místech, kde není potřebná přesnost nebo důkladnější analýza procházejících dat, ale spíš jde o vysokorychlostní přenosy velkých množství dat.

Nevýhodou je nízká úroveň kontroly procházejících spojení, která zejména u složitějších protokolů (např. FTP, video/audio streaming, RPC apod.) 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.

Mezi typické představitele paketových filtrů patří např. tzv. **ACL** (Access Control Lists) ve starších verzích operačního systému IOS na routerech spol. Cisco Systems, popř. JunOSspol. Juniper Networks, starší varianty firewallu v linuxovém jádře (ipchains).

Stavový firewall

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. Je schopen rozlišovat různé stavy paketů v rámci jednotlivých relací (spojení) a jeho úkolem je propustit pouze takové, které patří do již povolené relace (jiné jsou zamítnuty).

Obdobou stavového firewallu je ***nestavový firewall***, který se rozhoduje pouze na základě informací obsažených v konkrétním paketu (pracuje na nižší síťové vrstvě ISO/OSI modelu) a ***aplikační firewall,*** který pracuje naopak na vyšší síťové vrstvě.

Klasický příklad, kdy může provoz sítě selhat s nestavovým firewallem, je **FTP** (File Transfer Protocol). Tento protokol dle návrhu může pracovat ve dvou režimech. V aktivním režimu klient odešle (na port 21) serveru číslo portu (větší než 1024) a server se na něj ze svého portu (20) připojí. V pasivním režimu to funguje přesně opačně – server pošle klientovi port (větší než 1024) a on se na něj připojí (z portu většího než

1024). Preferovanější režim je zpravidla pasivní, díky vyšší bezpečnosti. Problém spočívá v tom, že pracovní port se mění s každým připojením, takže není možné napsat bezpečné a jednoznačné statické filtrovací pravidlo, které by dokázalo FTP spojení rozpoznat. Stavové firewally tento problém řeší tím, že si udržují tabulku navázaných spojení a inteligentně asociují nové požadavky na připojení s existujícími v tabulce. Například v Linuxu modulip_conntrack_ftp dokáže z navazovaného spojení „odposlechnout" kombinaci portů. Dále pak zajistí, že stavový firewall tyto pakety označí RELATED – tedy povolené.

Šifrování a autentizace

Základní pojmy

Pojmem **utajení** (angl. confidentality) rozumíme přenos dat takovým způsobem, že cizí posluchač naslouchající na přenosovém kanále významu přenášených dat nerozumí.

Autentizace (angl. authentication) zdroje dat dává příjemci jistotu, že odesílatel dat je skutečně tím, za koho se vydává. Zajištěná integrita dat (data integrity) dává příjemci jistotu, že data nebyla na cestě nikým zmodifikována.

Kryptografickým systémem rozumíme systém, který na straně odesílatele dat šifruje zprávu, přenáší ji zašifrovanou (angl. cyphertext) a na straně příjemce původní zprávu dešifruje. Šifrování lze principiálně realizovat dvěma způsoby.

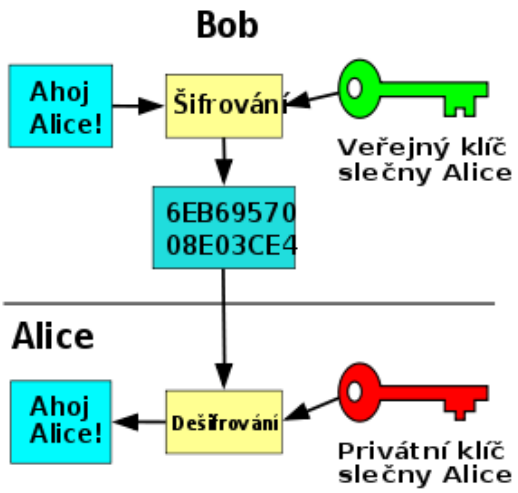
- Prvním z nich je vyvinout konkrétní šifrovací algoritmus a ten přede všemi mimo příjemce utajit. V případě prozrazení je však (možná i hardwarová) implementace takovéhoto systému k ničemu.
- Proto se zpravidla používá druhý způsob a to použít (i veřejně známý) algoritmus, jenž však bude parametrizován pomocí klíčů, které jediné musí zůstat utajeny. Podmínkou je pouze to, aby bylo dost možných klíčů, aby útočník nebyl schopen jednoduše vyzkoušet všechny z nich.

Slovem **šifra** nebo šifrování budeme označovat kryptografický algoritmus, který převádí čitelnou zprávu neboli prostý text na její nečitelnou podobu neboli šifrový text. **Klíč** je tajná informace, bez níž nelze šifrový text přečíst. **Hašovací funkce** (speciálně navržena pro kryptografii) je způsob, jak z celého textu vytvořit krátký řetězec, který s velmi velkou pravděpodobností identifikuje nezměněný text. Certifikáty a elektronický podpis jsou softwarové prostředky, které umožní šifrování textu.

Základní typy šifrování

S ohledem na způsob zacházení s klíči rozlišujeme v kryptografii 2 způsoby šifrování:

- **Symetrická šifra** je taková, která pro šifrování i dešifrování používá tentýž klíč a to jak na vysílači, tak na přijímači.
- **Asymetrická šifra** používá veřejný klíč pro šifrování a soukromý klíč pro dešifrování, který zná pouze příjemce zprávy. Asymetrická šifra má oproti symetrické tu výhodu, že ji odpadá problém s utajenou distribucí klíčů. Na druhou stranu používané algoritmy pro šifrování jsou mnohem složitější a náročnější na výpočty (např. RSA, Diffie-Hellman) a tudíž pomalejší. Uplatnění má např. u digitálního podpisu. ***Na obrázku**: Bob chce poslat Alici zašifrovanou zprávu - zašifruje ji veřejným klíčem Alice a ona už si ho rozšifruje svým soukromým.*



Šifrování v symetrickém i asymetrickém systému je realizováno konkrétním šifrovacím algoritmem. Podívejme se ale nyní, jak lze v obou těchto systémech prakticky zrealizovat autentizaci a na ní se vážící zabezpečení integrity dat.

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.

Autentizace na základě symetrického šifrování (sdíleného klíče) a otisku (hashe)

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 .

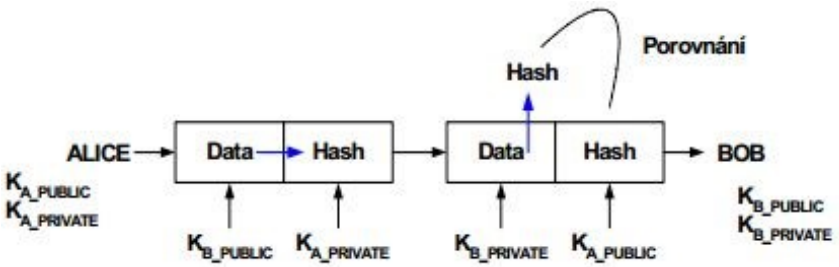
Pro výpočet otisku používáme vhodnou jednosměrnou funkci (hash), která z velkého bloku dat vypočte tzv „otisk“ – blok několika bajtů, jenž blok dat charakterizuje. Hash funkce je volena tak, aby nebylo možné najít a podvrhnout jiný text zprávy, který bude mít stejný otisk jako zpráva původní.

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.

Autentizace na základě asymetrického šifrování (sdíleného klíče) a otisku (hashe)

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.

Virtuální privátní sítě (VPN)

VPN (***Virtual Private Network***) je prostředek pro propojení několika počítačů na různých místech internetu do jediné virtuální počítačové sítě. I když počítače mohou být v naprosto fyzicky nezávislých sítích na různých místech světa, prostřednictvím virtuální privátní sítě mezi sebou mohou komunikovat, jako by byly na jediném síťovém segmentu.

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.

Zobecněním VPN je síťové tunelování, kdy se prostřednictvím standardního síťového spojení vytvoří virtuální linka mezi dvěma počítači, v rámci které pak lze navázat další síťová spojení.

VPN s IPSec

Řešení VPN využívajících IP (IP VPN) je pro zabezpečení nejčastěji založeno na IPSec (Internet Protocol SECurity). Ten jako komplexní soubor protokolů na síťové vrstvě nabízí tunelování, šifrování a autentizaci. Uzly, které spolu chtějí komunikovat s využitím IPSec, se musí nejprve dohodnout na

bezpečnostní politice ve formě bezpečnostní asociace. Tunel mezi dvěma servery nebo mezi serverem a uživatelem pak zabezpečuje provoz jakéhokoli typu.

IPSec představuje vysoce bezpečnou metodu pro budování VPN, ale je současně také složitou metodou na implementaci.

VPN s SSL

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 instalace. SSL ale nepodporuje všechny aplikace: hodí se pro zabezpečení webové komunikace.

IPsec

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

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.

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 v 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.