IV. Počítače a sítě

Update: 5. května 2018

Obsah

1 Architektura univerzálních procesorů. Principy urychlování činnosti procesorů.

Architektura počítačů je náčrt struktury a funkčnosti systému. Je charakterizována výčtem **registrů** a jejich funkcí, vnitřních a vnějších **sběrnic**, způsobem **adresování** a **instrukčním souborem**.

Registr je malé úložiště 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.

1.1 Procesory CISC a RISC

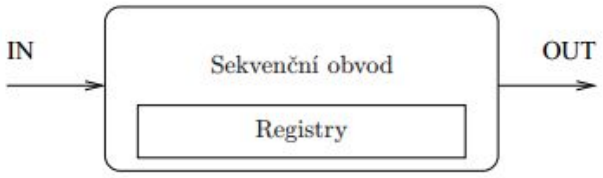
V dnešní době se ustálilo dělení počítačů do dvou základních kategorií podle typu používaného procesoru:

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

1.1.1 CISC

- procesory s **komplexním instrukčním souborem**
- instrukce mají **proměnlivou délku** i **dobu vykonání**
- **vysoká složitost instrukcí** → nutný systematický návrh řadiče procesoru
- vykonání strojové instrukce probíhá posloupností mikrooperací (předepsána mikroinstrukcí v řídící paměti)
- procesor obsahuje relativně **nízký počet registrů**
- operace provedená i **složenou instrukcí** (např. násobení) může být **nahrazena** sledem jednodušších strojových instrukcí (*sčítání a bitové posuny*) → mohou být ve výsledu vykonány rychleji, než hardwarově implementovaná složená varianta
- Označení CISC bylo zavedeno jako **protiklad** až poté, co se prosadily procesory RISC, které mají instruční 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 jen jednu strojovou instrukci pro danou operaci (např. *logické 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

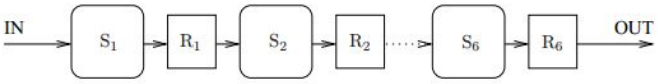
Krok		Význam
1.	**VI**	Výběr Instrukce
2.	**DE**	Dekódování
3.	**VA**	Výpočet Adresy
4.	**VO**	Výběr Operandu
5.	**PI**	Provedení Instrukce
6.	**UV**	Uložení Výsledku

	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

1.1.2 RISC

- počet instrukcí a způsobů adresování je malý, ale zůstává úplný, aby bylo možno provést vše → v tomhle se liší od CISC
- 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 problémů s frontou instrukcí
- mikroprogramový řadič může být nahrazen rychlejším obvodem
- přenášejí složitost technologického řešení do programu (překladače)
- představitelé *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

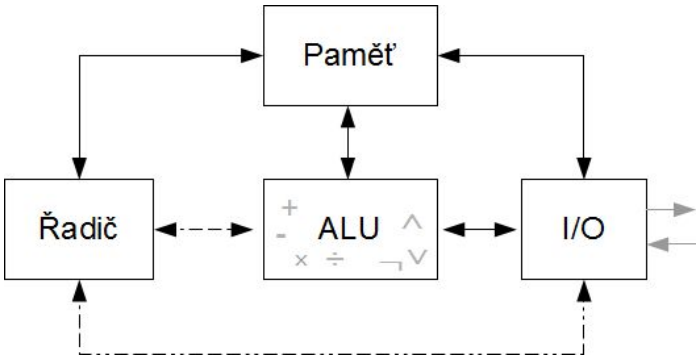
Tabulka 5: Zřetězené provádění instrukcí procesorem RISC

1.2 Von Neumannovo schéma počítače

John Von Neumann definoval v roce **1945** základní koncepci počítače (EDVAC) **řízeného obsahem paměti**. Od té doby se objevilo několik odlišných modifikací, ale v podstatě se **počítače v dnešní době** konstruují podle tohoto modelu. Ve svém projektu si von Neumann stanovil určitá kritéria a principy, které musí počítač splňovat, aby byl použitelný univerzálně. Můžeme je ve stručnosti shrnout do následujících bodů:

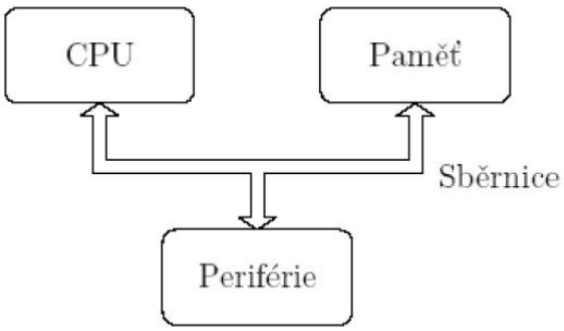
- Počítač se skládá z paměti, řídící jednotky, aritmeticko–logické jednotky, vstupní a výstupní jednotky.
  - **ALU** - aritmeticko–logická jednotka (aritmetic–logic unit) - jednotka provádějící veškeré aritmetické výpočty a logické operace. Obsahuje sčítačky, násobičky a komparátory.
  - **Operační paměť** - slouží k uchování zpracovávaného programu, zpracovávaných dat a výsledků výpočtu
  - **Řídící jednotka** - řídí činnost všech částí počítače. Toto řízení je prováděno pomocí řídících signálů, které jsou zasílány jednotlivým modulům. Řadiči jsou pak zpět zasílané stavové hlášení. Dnes řadič spolu s ALU tvoří jednu součástku, a to procesor neboli CPU (Central Processing Unit).
  - **Vstup/ Výstup** – zařízení určené pro vstup dat, a výstup zpracovaných výsledků.
- Struktura pc je **nezávislá na typu řešené úlohy** (univerzálnost), počítač se programuje obsahem paměti.
- Následující krok počítače je závislý na kroku předešlém.
- **Instrukce** a **data** jsou v téže paměti.
- Paměť je rozdělena do **paměťových buněk stejné velikosti (Byte)**, jejichž pořadová čísla se využívají jako adresy.
- Program je tvořen posloupností instrukcí, které se vykonávají jednotlivě v pořadí, v jakém jsou zapsány do paměti.

- Změna pořadí prováděných instrukcí se provádí **skokovými instrukcemi** (podmíněné nebo nepodmíněné skákání na adresy).
- Čísla, instrukce, adresy a znaky se značí v **binární soustavě**.



Nevýhody Von Neumannovy koncepce ve srovnání s dnešními PC

- Podle von Neumannova schématu počítač pracuje **vždy nad jedním programem**. Toto vede k velmi špatnému využití strojového času. Dnes je obvyklé, že počítač **zpracovává paralelně více programů** zároveň - tzv. **multitasking**
- Počítač může mít i více jak jeden procesor.
- Podle Von Neumanova schématu mohl počítač pracovat pouze v tzv. **diskrétním režimu**, kdy byl do paměti počítače zaveden program, data a pak probíhal výpočet. V průběhu výpočtu již nebylo možné s počítačem dále interaktivně komunikovat.
- Dnes existují **vstupní/výstupní** zařízení, např. pevné disky a páskové mechaniky, které umožňují vstup i výstup.
- Program se do paměti nemusí zavést celý, ale je možné zavést pouze jeho část a ostatní části zavádět až v případě potřeby.



Výhody

+ **Rozdělení paměti** pro kód a data určuje programátor, řídící jednotka přistupuje pro data i instrukce jednotným způsobem.

+ **Jedna sběrnice** -> jednodušší levnější výroba.

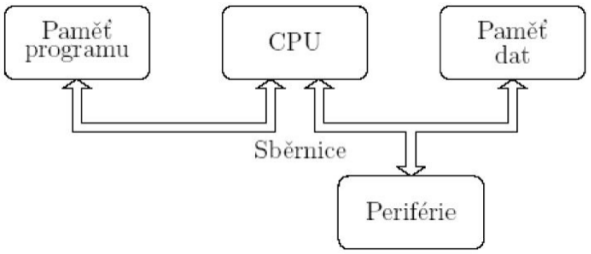
Nevýhody

– **Společné uložení dat a kódu** může mít za následek přepsání vlastního programu

– **Jedna sběrnice** je omezující.

1.3 Hardvardské schéma počítače

Několik let po von Neumannovi, přišel vývojový tým odborníků z Harvardské univerzity s vlastní koncepcí počítače, která se sice od Neumannovy příliš nelišila, ale odstraňovala některé její nedostatky. V podstatě jde pouze o **oddělení paměti pro data a program**. Abychom si mohli obě koncepce porovnat, můžeme vycházet ze zjednodušených schémat.



Výhody

+ **Program se nepřepíše** (oddělené paměti pro data a program).

+ Dvě sběrnice umožňují **paralelní** načítání instrukcí a dat.

+ Paměti mohou být vyrobeny **odlišnými technologiemi** a každá může mít jinou nejmenší adresovací jednotku (8 bitů pro instrukce a 8, 16 nebo 32 pro data).

Nevýhody

– 2 sběrnice mají **vyšší nároky na vývoj** řídící jednotky a jsou také dražší a složitější na výrobu.

– Paměť je **rozdělena** už od **výrobce**.

– Nevyužitou část dat **nelze využít** po program a obráceně.

1.4 Principy urychlování činnosti procesorů

Techniky urychlování výpočtu v hardwaru:

– speciální kódování dle potřeby dané úlohy

– speciální výpočetní jednotky dle potřeby dané úlohy (FFT – rychlý fourierova transformace)

– paralelní zpracování (násobné výpočetní jednotky)

– zřetězové zpracování instrukcí (pipelining)

1.4.1 Paralelní zpracování

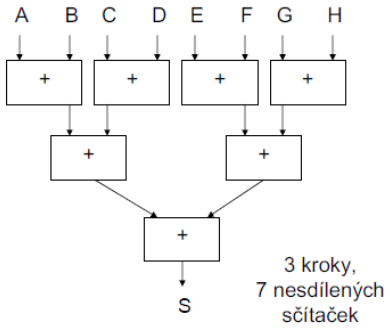
Zpracování více elementárních úloh běží současně.

Př. S = A + B + C + D + E + F + G + H

```
SW: sekvenčně

S = A + B;
S = S + C;
S = S + D;
S = S + E;
S = S + F;
S = S + G;
S = S + H;

7 kroků,
1 sdílená
sčítačka
```



1.4.2 Zřetězené zpracování instrukcí (pipelining)

Princip zřetězení se značně překrývá s principy procesorů RISC. Základní myšlenkou je **rozdělení zpracování jedné instrukce** mezi různé části procesoru a tím i dosažení možnosti **zpracovávat více instrukcí** najednou. Pro dosažení tohoto 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 instrukcí. 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 standartem 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 (1bit)**, 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 paměť. V případě, že se predikce nepovede, bývá nutné vyprázdnit celou pipeline a začít vyzvedávat instrukce ze správné adresy, což znamená relativně **velké zdržení**. Související 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 instrukcí*)

- u malého zřetězení **neřešíme**
- používání bublin na vyprázdnění pipeline, **naplněníní prázdnými instrukcemi**
- **predikce skoku** – vyhrazen jeden bit předurčující, zda se skok provede či nikoliv
  - **Statická** – součást instrukce → řeší programátor nebo kompilátor
  - **Dynamická**
    - **jednobitová** – zaznamenává jestli se skok provedl, či ne (1/ 0)
    - **dvoubitová** – metoda zpožděného skoku → v procesoru řeší se např. tabulkou s 4 kB instrukcí

Zřetězené zpracování přináší urychlení výpočtu nejen v procesorech, ale i jiných číslicových obvodech (např. pro zpracování obrazu, bioinformatických dat apod.). Pokud použijeme zřetězené zpracování, musíme dodat řadu podpůrných obvodů a řešit řadu nových problémů. Moderní procesory používají kromě zřetězení i další koncepty:

- **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.
- **VLIW procesory** – má více ALU – tzn. může zároveň dělat více operací → k tomu složí dlouhé instrukce.
- **vektorové procesory** – je navržený tak, aby dokázal vykonávat matematické operace nad celou množinou čísel v daném čase. Je opakem skalárního procesoru, který vykonává jednu operaci s jedním číslem v daném čase.
- **multivláknové procesory**

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

Monolitické počítače (mikroprocesory)

- Mikroprocesory, mikrokontroléry, minipočítače jsou další názvy pro monolitické počíače.
- Jsou to malé počítače integrované v jediném pouzdře (all in one).
- Mají širokou oblast využití.
- Využívá se Harvardské koncepce, což umožňuje aplikovat paměti pro data a program různých technologií.
- Zjednodušené rysy architektury RISC.
- INTEL 8051 (standart), ATMEL, MICROCHIP PIC.
- V monolitických počítačích můžeme najít dva základní typy periférií (vstupní/výstupní).
- **Rozdělení pamětí**:
    - **pro data** – používáme většinou paměti energeticky závislé typu **RWM–RAM** (*Read–Write Memory–Random Access Memory*), tedy paměť s libovolným přístupem pro čtení i zápis. Jsou vyráběny jako statické (uchování paměti po celou dobu napájení), jejich paměťové buňky jsou realizovány jako klopné obvod.
    - **pro program** – se používají paměti typu **ROM** (*Read–Only Memory*) určené především ke čtění (paměť je uchována i po odpojení napájení). Mezi nejčastěji používané paměti patří **EPROM**, **EEPROM** (*Electrically Erasable Programmable Read-Only Memory*), **PROM** (*Programmable Read Only Memory*) a **Flash**.

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. Nejrychlejší.
- **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. Nejpomalejší.

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 jde o ochranu **mechanickou**. Odolávat náhodným nárazům, nebo i trvalým vibracím nebo elemtromagnetickým vlivům z okolí. Pro odstranění chyb, které nastanou působením vnějších vlivů nebo chyby programátora, je v mikropočítačích implementován speciální obvod nazývaný **WATCHDOG** $\rightarrow$ provede reinicializaci mikropočítače pomocí vnitřního RESETu, například při zacyklení. Watchod (WDT) se řadí mezi elektrické ochrany, mezi které můžeme zařenit **BROWN-OUT** – ochrana proti podpěti.

Typické periferie

Periferie - obvody, které zajišťují komunikaci mikropočítace s okolím.

1. **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**. 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.

2. **Čítace a časovače** - Do skupiny nejpoužívanějších periférií mikropočítače určitě patří čítače a časovače. *Časovač* se od **čítače** příliš neliší. Není, ale **inkrementován vnějším 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. Při přetečení časovače se i zde může automaticky předávat signál do přerušovacího podsystému mikropočítače.

3. **Sériové linky**: Sériový přenos dat je v praxi stále více používán. Dovoluje efektivním způsobem přenášet data na relativně velké vzdálenosti při 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**

4. **A/D a D/A 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.

5. **Obvody reálného času** (RTC - Reak Time Clock) - V mnoha aplikacích s použitím mikropočítačů je potřeba dodržovat přesnou časovou souvislost řízených událostí. Jde tedy o řízení v reálném čase. Ne vždy, ale taková posloupnost dostačuje a je nutno pro potřebu řízení udržovat skutečný čas, tedy hodiny, minuty, sekundy a případně i zlomky sekund. Pro tyto účely slouží obvody **RTC**. Při jejich použití je obvykle nutné vyřešit dva základní problémy:

   - **záložní zdroj** - je třeba zajistit záložní zdroj pro udržení nepřetržité činnosti obvodu (může dojít k výpadku proudu a tak i k ztrátě skutečného času).
   - **čtení dat** - čas je hodnota neustále se měnící. Např. pokud zahájíme čtení hodnoty v čase 10:59:59, může se stát, že po přečtění prvních dvou hodnot, v našem případě hodin, se čas posune na 11:00:00 a čtění dalších hodnot bude neplatné (řešení technicky pomocnými registry v RTC obvodu, nebo vhodným programovým řešením).

I_2C

- Dvoudrátová, dvouvodičová sběrnice se sériovým přenosem.
- Obsahuje slave a master obvody.
- Lze propojit až 128 zařízení. (Master, slave)
- **Adresa zařízení**: skládá se ze 7 bitů (horní 4 určuje výrobce, dolní 3 jdou nastavit libovolně)
- **Signály** - SCL (synchronous clock), SDA (synchronous data)

3 Struktura OS a jeho návaznost na technické vybavení počítače.

Operační systém je zpravidla tvořen tzv. **jádrem** (kernel), **ovladači I/O 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.

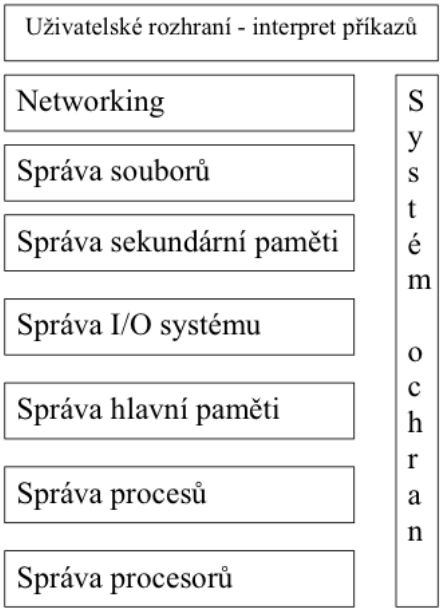
3.1 Jádro

Jádro se zpravidla dělí na dvě podstatné části:

1. **Správa procesů** – řeší problematiku aktivování a deaktivování procesů podle jejich priority resp. požadavků na prostředky (prakticky není u jednoduchých OS)
2. **Správa prostředků** – zajišťuje činnost I/O 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.

3.2 Generické komponenty OS

- Správa 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



3.2.1 Správa procesorů/procesů

Správce procesoru má tyto funkce:

- sleduje prostředek (procesor a stav procesů),
- rozhoduje, komu bude dána možnost užít procesor,
- přiděluje procesoru prostředek, tj. procesor,
- požaduje vrácení prostředku (procesoru).

Pojem **proces** (task) je nějaká činnost řízená programem. Proces potřebuje pro svou realizaci jisté zdroje:

- dobu procesoru,
- paměť,
- I/O zařízení, atd.

OS je z hlediska **správy procesů** zodpovědný za:

- vytváření a rušení procesů,
- potlačení a obnovení procesů,
- poskytnutí mechanismů pro synchronizaci procesů a pro komunikaci mezi procesy.

OS je z hlediska **správy procesorů** zodpovědný za výběr procesu běžícího na volném procesoru.

3.2.2 Správa (hlavní, operační) paměti

Jedná se o úložiště připravených tj. rychle dostupných dat sdílených procesorem a I/O zařízeními. Hlavní paměť je pole samostatně adresovatelných slov nebo bytů, zpravidla energeticky závislá.
OS je z hlediska správy **(hlavní) paměti** odpovědný za:

- vedení přehledů kdo a kterou část paměti v daném okamžiku využívá,
- rozhodování, kterému procesu uspokojit jeho požadavek na prostor paměti po uvolnění,
- přidělování a uvolňování paměti dle potřeby,
- řízení virtuální paměti.

Z hlediska těchto zodpovědností správce operační paměti:

- udržuje přehled o přidělované a volné paměti,
- ve spolupráci se správou procesů rozhoduje o tom, kterému procesu, kolik, kde a kdy má přidělit operační paměť,
- provádí přidělení volné části paměti,
- určuje strategii odnímání dříve přidělené operační paměti procesům (opět po předchozí domluvě se správou procesů).

3.2.3 Správa I/O systému

Správce periferních (I/O systému) má tyto funkce:

- sleduje stav prostředků (periferních zařízení, jejich řídících jednotek),
- rozhoduje o efektivním způsobu přidělování prostředku – periferního zařízení,
- přiřazuje prostředek (periferní zařízení) a zahajuje I/O operaci,
- požaduje navrácení prostředku.

Z hlediska funkce OS lze správce I/O systému chápat jako:

- úložiště vyrovnávacích pamětí,
- univerzální rozhraní ovladače I/O zařízení,
- ovladače jednotlivých hardwarových I/O zařízení.

Do správy I/O systému patří i **správa vnější (sekundární) paměti**. Počítačový systém musí poskytnout pro zálohování hlavní paměti sekundární paměť (HDD, SSD). OS je z hlediska správny vnější (sekundární) paměti odpovědný za:

- správu volné paměti,
- přidělování paměti,
- plánování činnosti disku.

3.2.4 Správa souborů

Správce souborů má tyto funkce:

- sleduje prostředek (soubor), jeho umístění, užití, stav atd.,
- rozhoduje, komu budou prostředky **přiděleny**, realizuje požadavky na **ochranu informací uložených** v souborech a realizuje operace přístupu k souborům,
- **přiděluje** prostředek, tj. otevírá soubor,
- **uvolňuje** prostředek, tj. uzavírá soubor.

Pod pojmem soubor chápeme jak programy, tak data. OS je z hlediska správy souborů odpovědný za:

- vytváření a rušení souborů,
- vytváření a rušení adresářů (katalogů, složek),
- podporu primitivních **operací** pro manipulaci se soubory a adresáři,
- archivování souborů na energeticky nezávislí média.

3.2.5 Networking, distribuované systémy

Distribuovaný systém je **kolekce procesorů**, které **nesdílejí** ani **fyzickou paměť** ani **hodiny**, synchronizující činnost procesoru. Každý procesor má svoji **lokální paměť** a **lokální hodiny**. Procesory distribuovaného systému jsou **propojeny** komunikační sítí. Komunikace jsou řízeny protokoly. Distribuovaný systém uživateli zprostředkovává přístup k různým zdrojům systému.

3.2.6 Systém ochran

Jsou to mechanismy pro **řízení přístupu** k **systémovým** a **uživatelským zdrojům**. Systém ochran musí:

- **rozlišovat** mezi **autorizovaným** a **neautorizovaným** použitím,
- specifikovat problém vnucovaného řízení,
- poskytnout prostředky pro své prosazení.

3.2.7 Uživatelské rozhraní - interpret příkazů

Interpret příkazů je program, umožnující vykonávat příkazy pro:

- správu a vytváření procesů – služby OS poskytované interperetem příkazů slouží k **provedení programu**, tj. k schopnosti OS zavést program do hlavní paměti a spustit jeho běh,
- ovládání I/O zařízení – uživatelský program **nesmí** provádět I/O operace **přímo**, OS musí poskytovat prostředky k provádění I/O operací,

- správu sekundární paměti – manipulace se systémem souborů, schopnost číst, zapisovat, vytvářet a rušit soubory,
- správu hlavní paměti,
- zpřístupňování souborů,
- ochranu – tj. d**etekci chyb v procesoru** a **paměti**, I/O zařízeních a v programech uživatelů pro zajištění správnosti výpočtu,
- práci v síti – **výměna informací** mezi procesy realizována buďto v rámci jednoho počítače nebo mezi různými počítači pomocí sítě, tj. implementace sdílenou pamětí nebo předávání zpráv.

Uživatelská rozhraní jsou realizovaná **znakově** (někdy označované řádkově) nebo **graficky**. Znakově orientovaným interpretům zadáváme příkazy pomocí klíčových slov, graficky orientovaným pomocí poklepání myši nebo dotykem na dotykové obrazovce na ikonu, pomocí dialogů apod.

3.2.8 Vnitřní služby operačního systému

Vnitřní služby OS **nejsou** určeny k tomu, aby **pomáhaly uživateli**, v prvé řadě slouží pro **zabezpečení efektivního provozu systému**, tj. slouží pro:

- Přidělování prostředků (zdrojů) mezi více souběžně operujících uživatelů nebo úloh.
- Účtování a udržování přehledu o tom, kolik jakých zdrojů systému který uživatel používá. Cílem je účtování za služby a sběr statistik pro plánování.
- Ochranu tj. péči o to, aby veškerý přístup k systémovým zdrojům byl pod kontrolou.

Vnitřní služby OS jsou obecně **realizovány souborem systémových programů** vytvářejících určité systémové struktury tzv. virtuální stroje. Typickými službami jsou programy pro:

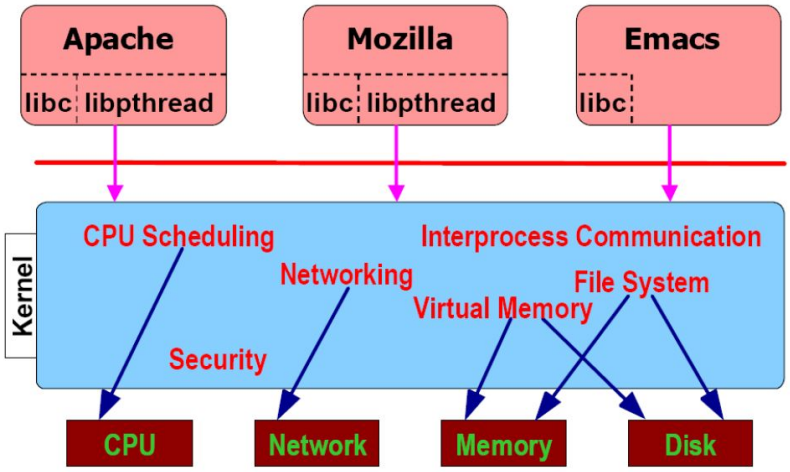
- práci se soubory, editaci souborů, katalogizaci souborů, modifikaci souborů,
- získávání, definování a údržbu systémových informací,
- podporu jazykových prostředí,
- zavádění a provádění programů,
- komunikace a řízení aplikačních programů.

3.3 Struktura OS podle jádra

- Monolitická jádra (monolithic kernel)
- Otevřené systémy (Open oystems)
- Microkernel

3.3.1 Monolitický OS

OS je na jedno místě (v obrázku pod „červenou čarou“). Aplikace používá dobře definované rozhraní systémového volání pro komunikaci s jádrem.



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

Výhody

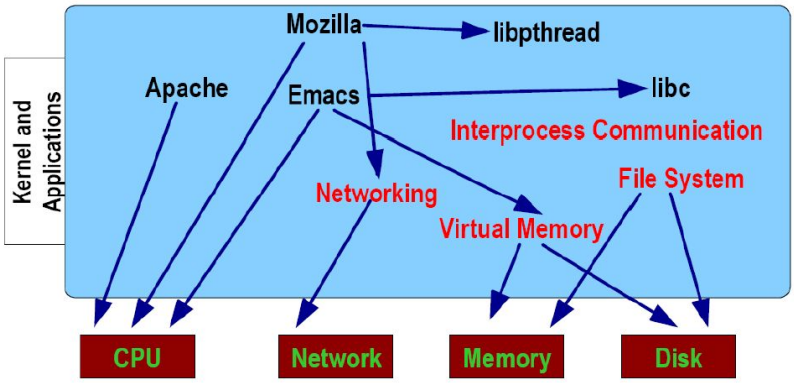
+ dobrý výkon
+ dobře pochopená koncepce
+ jednoduché pro vývojáře jádra
+ vysoká úroveň ochrany mezi aplikacemi

Nevýhody

– žádná ochrana mezi komponentami jádra
– není jednoduše a bezpečně rozšiřitelné jádro
– celková struktura je ve výsledku komplikovaná, neexistující hranice mezi moduly jádra

3.3.2 Otevřené Systémy

Aplikace, knihovny i jádro jsou všechny v jednom adresovaném prostoru.
Příklady OS – 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

3.3.3 Microkernel OS

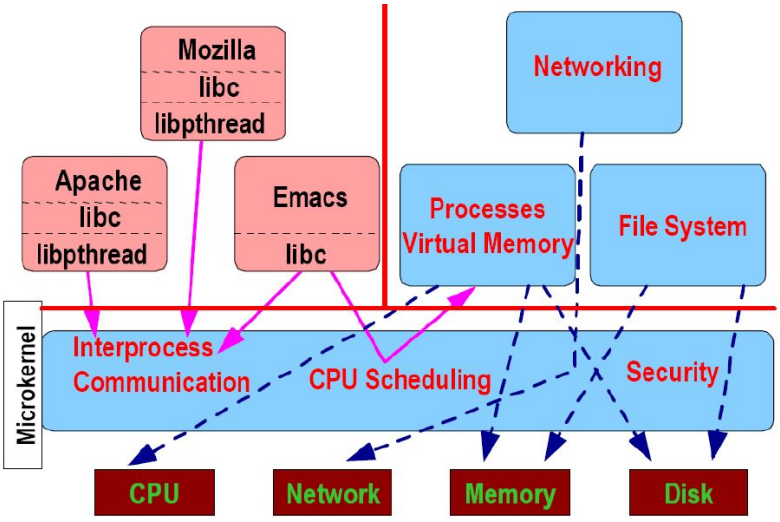
Filosofie návrhu – chráněné jádro obsahuje pouze minimálné (malou, čistou, logickou) množinu abstrakcí:

- procesy a vlákna
- virtuální paměť
- komunikace mezi procesy

Všechno ostatní jsou server–procesy běžící na uživatelské úrovni.

Příklady OS – Marc, Chorus, QNX, GNU Hurd

Zkušenosti s tímto návrhem jsou smíšené.



Výhody

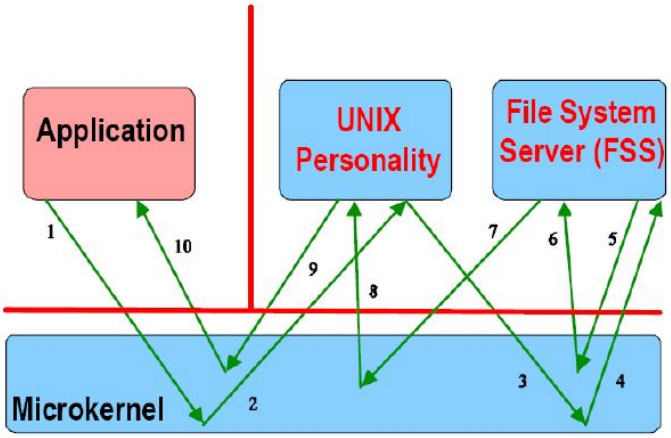
+ přidáním server–procesu se rozšíří funkcionalita OS
+ jádro nespecifikuje prostředí OS
+ servery na uživatelské úrovni se nemusí zabývat hardwarem
+ silná ochrana OS i sám proti sobě (části OS jsou oddělené servery)
+ jednoduché rozšíření na distribuovaný nebo multiprocesorový systém

Nevýhody

– výkon (systémová volání, viz příklad pod touto kapitolou)
– špatná minulost

3.3.4 Příklad systémového volání

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.

Model ISO/OSI

Počítačové sítě vyvíjelo více firem, zpočátku to byly uzavřené a nekompatibilní systémy. Hlavním účelem sítí je však vzájemné propojování, a tak vyvstala potřeba stanovit pravidla pro přenos dat v sítích a mezi nimi. **Mezinárodní ústav pro normalizaci ISO** (International Standards Organization) vypracoval tzv. referenční model **OSI (Open Systems Interconnection)**, který rozdělil práci v síti do **7 vzájemně spolupracujících vrstev**.

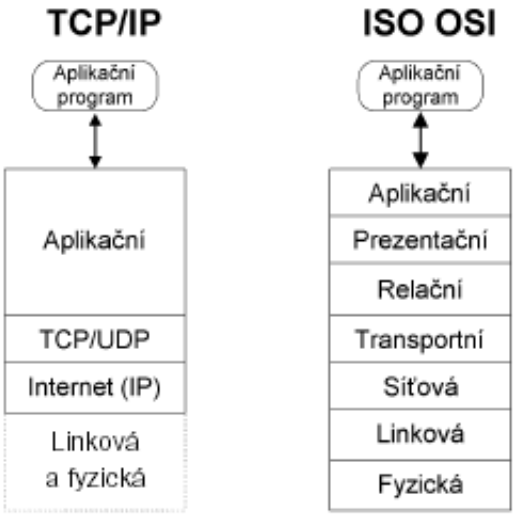
Jak již bylo řečeno, model ISO/OSI rozděluje síťovou práci na vrstvy. Princip spočívá v tom, že vyšší vrstva převezme úkol od podřízené vrstvy, zpracuje jej a předá vrstvě nadřízené. Vertikální spolupráce mezi vrstvami (nadřízená s podřízenou) je **věcí výrobce** sítě. Model **ISO/OSI doporučuje**, jak mají vrstvy **spolupracovat horizontálně** – dvě stejné vrstvy modelu mezi různými sítěmi (či síťové prvky různých výrobců) musejí spolupracovat. Model je důležitý především pro výrobce síťových komponent. V praktické práci se sítí jej moc nevyužijeme. Umožňuje však pochopit principy práce síťových prvků a zároveň patří k základní terminologii sítí.

- **Aplikační vrstva** - Je **určitou aplikací** (např. oknem v programu) zpřístupňující uživatelům síťové služby. Nabízí a zajišťuje přístup k souborům (na jiných počítačích), vzdálený přístup k tiskárnám, správu sítě, elektronické zprávy (včetně e-mailu)...
- **Prezentační vrstva** - Má na starosti **konverzi dat**, přenášená data mohou totiž být v různých sítích různě kódována. Tato vrstva zajišťuje sjednocení formy vzájemně přenášených údajů. Dále data komprimuje, případně šifruje... V praxi často splývá s relační vrstvou.
- **Relační vrstva** - **Navazuje** a po skončení přenosu **ukončuje spojení**. Může provádět **ověřování** uživatelů, **zabezpečení** přístupu k zařízením...
- **Transportní vrstva** - Tato vrstva **zajišťuje přenos dat mezi koncovými uzly**. Jejím účelem je poskytnout takovou kvalitu přenosu, jakou požadují vyšší vrstvy. Vrstva nabízí spojově (TCP) a nespojově orientované (UDP) protokoly. (Platí pouze pro **TCP/IP**)
- **Síťová vrstva** - Je zodpovědná za spojení a **směrování mezi dvěma počítači nebo celými sítěmi** (tj. uzly), mezi nimiž neexistuje přímé spojení. Stará se o síťové adresování.
- **Linková (spojová) vrstva** - Poskytuje **spojení mezi dvěma sousedními systémy**. **Uspořádává data** z fyzické vrstvy do logických celků známých jako **rámce** (frames). Seřazuje přenášené rámce, stará se o nastavení parametrů přenosu linky, oznamuje neopravitelné chyby. Formátuje fyzické rámce, opatřuje je fyzickou adresou a poskytuje synchronizaci pro fyzickou vrstvu.

- **Fyzická vrstva** - Fyzická vrstva definuje všechny elektrické a fyzikální vlastnosti zařízení. Jakým signálem je reprezentována logická jednička, jak přijímací stanice rozezná začátek bitu, jaký je tvar konektoru, k čemu je který vodič v kabelu použit.

Rodina protokolů TCP/IP (Transmission Control Protocol/Internet Protocol) obsahuje **sadu protokolů** pro komunikaci v počítačové síti a je hlavním protokolem celosvětové sítě **Internet**. Komunikační protokol je množina pravidel, která určují syntaxi a význam jednotlivých zpráv při komunikaci. Architektura TCP/IP je členěna do čtyř vrstev (na rozdíl od referenčního modelu OSI se sedmi vrstvami):

1. **Vrstva síťového rozhraní** (Network interface)
2. **Síťová (IP) vrstva** (Internet layer)
3. **Transportní vrstva** (Transport layer)
4. **Aplikační vrstva** (Application layer)



Komunikace mezi **stejnými vrstvami dvou různých systémů** je řízena **komunikačním protokolem** za použití spojení vytvořeného sousední nižší vrstvou. Architektura umožňuje výměnu protokolů jedné vrstvy bez dopadu na ostatní. Příkladem může být možnost komunikace po různých médiích fyzické vrstvy modelu OSI - ethernet (optické vlákno, kroucená dvojlinka, Wi-Fi), sériová linka.

1. Vrstva síťového rozhraní

Nejnižší vrstva umožňuje **přístup k fyzickému přenosovému médiu**. Je specifická pro každou síť v závislosti na její implementaci. Příklady sítí: Ethernet, Token ring, FDDI, 100BaseVG, X.25, SMDS.

2. Síťová vrstva

Vrstva zajišťuje p**ředevším síťovou adresaci**, **směrování** a předávání datagramů (**packety**). Protokoly: **IP**, **ARP**, **RARP**, **ICMP**, **IGMP**, **IGRP**, **IPSEC**. Je implementována ve všech prvcích sítě - směrovačích i koncových zařízeních.

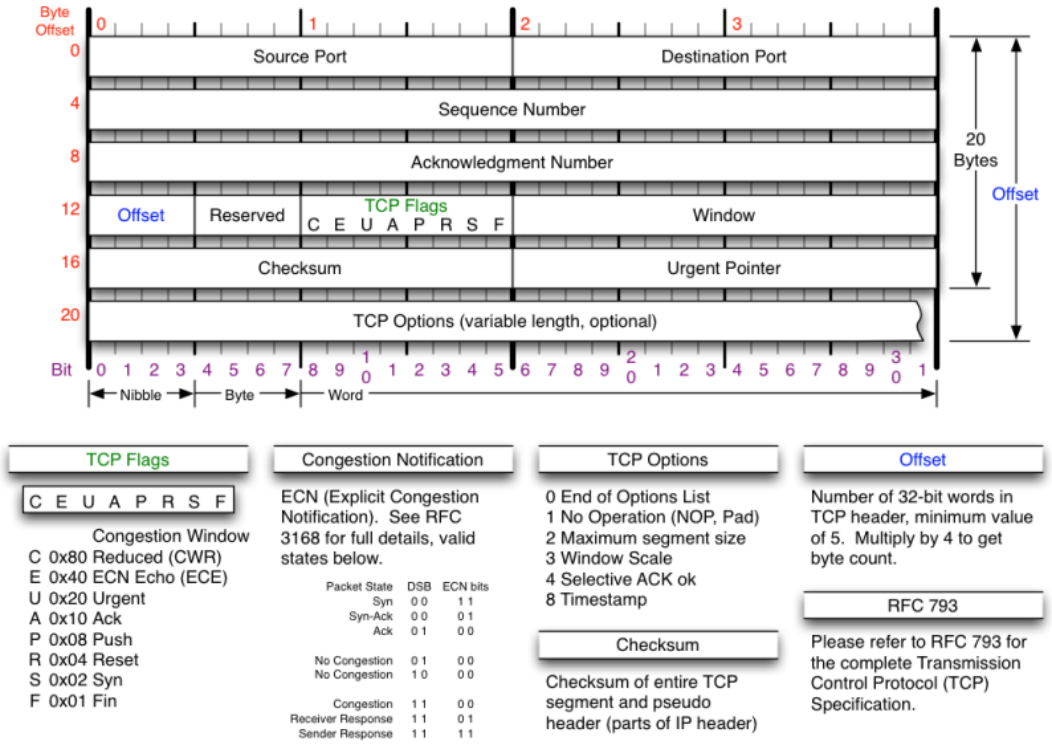
Protokol IP (Internet Protocol). Od nadřazených protokolů transportní vrstvy obdrží datové segmenty s požadavkem na odeslání. K segmentům připojí vlastní hlavičku a vytvoří IP datagram. V IP hlavičce je především IP adresa příjemce a odesílatele. IP protokol je **nespojový** (před zahájením výměny dat nevytváří relaci) a **nespolehlivý** (předání paketů na místo určení není kontrolováno). Paket IP se tedy může ztratit, být doručen mimo pořadí, zdvojen nebo zpožděn. Protokol IP neobsahuje prostředky pro zotavení z chyb tohoto typu. To vše má zajistit nadřízená transportní vrstva – protokol **TCP**.

3. Transportní vrstva

Transportní vrstva je implementována až v **koncových zařízeních** (počítačích) a umožňuje proto přizpůsobit chování sítě potřebám aplikace. Poskytuje transportní služby kontrolovaným spojením spolehlivým protokolem **TCP** (transmission control protocol) nebo nekontrolovaným spojením nespolehlivým protokolem **UDP** (user datagram protocol).

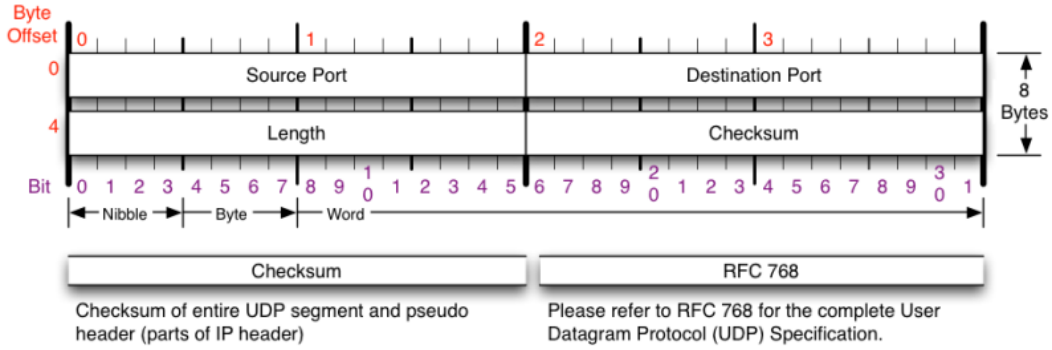
Protokol TCP (Transmission Control Protocol) vytváří **virtuální okruh** mezi koncovými aplikacemi, zajišťuje tedy spolehlivý přenos dat.

- Spolehlivá transportní služba, doručí adresátovi všechna data **bez ztráty** a ve **správném pořadí**.
- Služba se spojením, má fáze navázání spojení, přenos dat a ukončení spojení.
- Transparentní přenos libovolných dat.
- **Plně duplexní spojení**, současný obousměrný přenos dat.
- Rozlišování aplikací pomocí portů.
- 3-way handshake
- Komunikace je řízená pomocí příznakových bitů (ACK, SYN, atd.)



Protokol UDP (User Datagram Protocol) poskytuje **nespolehlivou** transportní službu pro takové aplikace, které nepotřebují spolehlivost, jakou má protokol TCP. Nemá fázi navazování a ukončení spojení a už první segment UDP obsahuje aplikační data. UDP je používán aplikacemi jako je **DHCP**, **TFTP**, **SNMP**, **DNS** a **BOOTP**.

- Nespolehlivá transportní služba, neověřuje zda data došla v pořádku nebo ve správném pořadí.
- Nižší režie než u TCP (rychlejší).
- Zajištění spolehlivosti je na aplikacích vyšší vrstvy.
- Nemá fázi navázání a ukončení spojení, rovnou zasílá data.
- Hlavička UDP má pouze 4 části (délku, zdrojový/cílový port, checksum)



4. Aplikační vrstva

Jedná se přímo o programy (procesy), které využívají přenosu dat po síti ke konkrétním službám pro uživatele. Příklady: **Telnet** (TCP 23), **FTP** (TCP 20, 21), **HTTP** (TCP 80), **DHCP** (UDP 67, 68), **DNS** (TCP/UDP 53), **SSH** (TCP 22).

Aplikační protokoly používají vždy jednu ze dvou základních služeb transportní vrstvy: **TCP** nebo **UDP**, případně obě dvě (např. DNS). Pro rozlišení aplikačních protokolů se používají tzv. **porty**, což jsou domluvená číselná označení aplikací. Každé síťové spojení aplikace je jednoznačně určeno **číslem portu** a **transportním protokolem** (a samozřejmě adresou počítače).

Překlad síťových adres NAT

NAT (Network address translation) se většinou používá pro přístup více počítačů z **lokální sítě** na **Internet** pod **jedinou veřejnou adresou**. Překládá zdrojovou a cílovou IP adresu, je realizován na **routerech**, **firewallech** většinou zařízeních 3 vrstvy. Umožňuje připojit více počítačů na jednu veřejnou IP adresu - řeší se tak nedostatek přidělených veřejných IP adres. Využívá se překladové tabulky.

Výhody:

- Zvyšuje bezpečnost počítačů připojených za NATem (potenciální útočník nezná opravdovou IP adresu).
- Umožňuje připojit více počítačů na jednu veřejnou IP adresu, čímž se obchází nedostatek IPv4 adres

Nevýhody:

- Zařízení za NATem nemají skutečné připojení k Internetu (není možné se snadno připojit na zařízení za NATem.)
- NAT znemožňuje správnou funkcionalitu některých software

Typy NATu

Typicky se využívá kombinace obou níže zmíněných řešení.

- **Statický NAT** - Překladová tabulka je konfigurována manuálně administrátorem.
- **Dynamický NAT** - Obsah překladové tabulky je vytvářen dynamicky v závislosti na síťovém provozu. Veřejné adresy sou alokovány jednotlivým spojením jejich vypůjčením z **NAT Poolu**.
- **Network adress and por translation NAPT** - Několik uzlů využívá pouze jednu veřejnou IP adresu. Jednotlivé uzly jsou identifikovány pomocí různých čísel portů.

IPv6

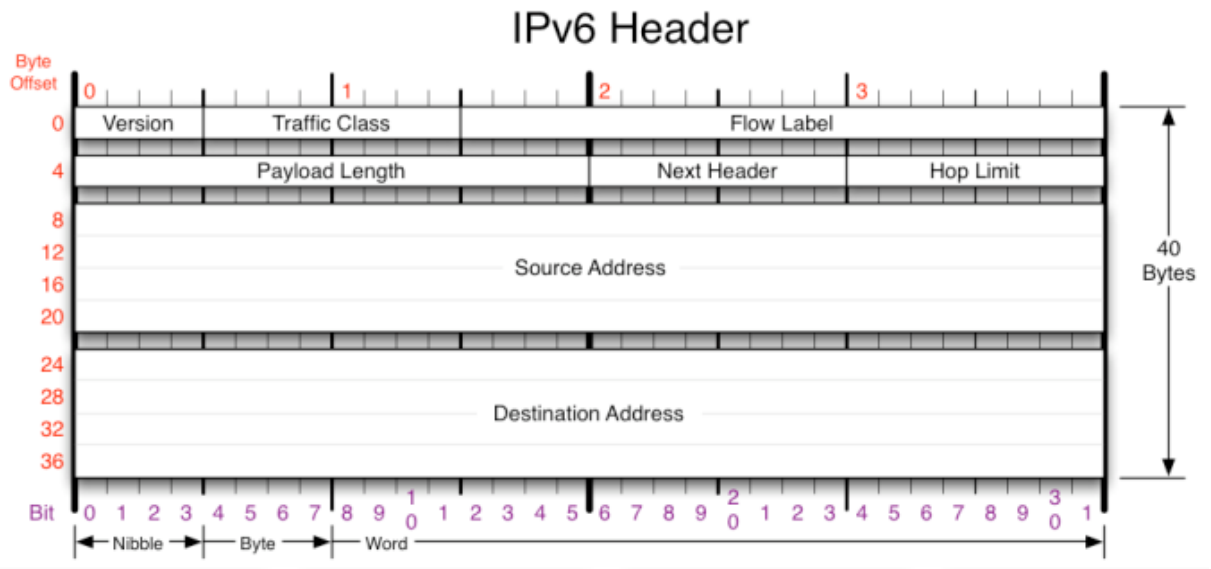
IPv6 **nahrazuje** dosluhující protokol IPv4. Přináší zejména **masivní rozšíření adresního prostoru** a zdokonalení schopnosti přenášet **vysokorychlostně data**. Starší protokol IPv4 poskytuje omezený adresní prostor – teoreticky 2^32 adres. IPv6. Obsahuje celkem 2^128 adres. Většina přenosových a aplikačních vrstev protokolů vyžaduje malé nebo žádné změny pro funkčnost s IPv6. Výjimkami jsou protokoly aplikací zahrnující adresy síťové vrstvy např.: FTP. **Multicast** je součástí základní specifikace IPv6 na rozdíl od IPv4, kde byl zaveden později. IPv6 nepoužívá **broadcast** na místní linku. Každá adresa má **128b**. **Odstraněna potřeba NAT**. Protokol pro IP vrstvu šifrování a autentizaci **IPsec** je integrální součástí souboru protokolů IPv6, na rozdíl od IPv4, kde je přítomen volitelně (obvykle ale implementován).

IPv6 adresy se obvykle zapisují jako osm skupin čtyř hexadecimálních číslic: **2001:0db8::1428:57ab**. Vzhledem k zdlouhavému zápisu se může nejdelší sekvence nul nahradit **::**. 4 nuly můžeme nahradit jednou.

IPv6 Packet

Paket IPv6 se skládá ze dvou hlavních částí: hlavičky a těla.

- **Verzi** - 4 bity, verze 6
- **Dopravní třídu** - 8 bitů na prioritu paketu. Úroveň priority se dělí na rozsahy: kde zdroj podporuje kontrolu přetížení a bez podpory kontroly přetížení.
- **Pojmenování toku** - 20 bitů pro správu QoS. Původně určeno pro speciální obsluhu aplikací reálného času, nyní se nepoužívá.
- **Délka těla** - 16 bitů pro délku těla paketu. Při vynulování se nastaví „jumbo“ tělo (skok za skokem)
- **Následující hlavička** - 8 bitů, určuje další vnořený protokol. Hodnoty se shodují s hodnotami definovanými pro IPv4.
- **Zdrojová a cílová adresa** - 128 bitů na každou adresu.
- **Hop limit** - 8 bitů, číselně definuje počet povolených přechodů síťovými prvky. Každý přechod znamená snížení čísla o 1.


IPv6 Header
Byte Offset
0
4
8
12
16
20
24
28
32
36
Version
Traffic Class
Flow Label
Payload Length
Next Header
Hop Limit
Source Address
Destination Address
40 Bytes
Bit
Nibble
Byte
Word

5 Metody sdíleného přístupu ke společnému 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 na náhodě (náhodné časové prodlevy) a musí řešit kolize (situace, kdy chce naráz na kanálu vysílat více stanic)

5.1 Bezkolizní (deterministické) metody sdílení média

Deterministické metody se dělí na:

- **Centralizované řízení** – jedna ze stanic je master a přiděluje ostatním právo vysílat. Efektivní, ale část kanálu použita pro komunikaci s masterem.
  - **Přidělování na výzvu** - nejstarší, původně v terminálových systémech nad protokoly BSC a HDLC.
  - **Cyklická výzva** - vyzývaná stanice **buď vyšle rámec, nebo odmítne výzvu**, příp. neodpoví. Použitelné pro **malé zpoždění** na kanále. Rozumné pro vysoké využití kanálu. Pro malé zatížení a velký počet stanic neefektivní.
  - **Binární vyhledávání** - pro kanál, u kterého může stanice rozpoznat, **zda vysílá jedna nebo více stanic**. Při malém zatížení a velkém počtu stanic je efektivní vyhledávat stanici připravenou vysílat binárním vyhledáváním. Stanice se zorganizují do stromu podle jednotlivých bitů adres, řídící stanice postupně vyzývá skupiny stanic, aktivní stanice ve vyzvané skupině 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. **Rychlejší pro malé zátěže.**
  - **Přidělování na žádost** - žádosti od stanic přicházejí k řídící stanici po **vyhrazených kanálech**, vyhrazených kanálů se v klasických počítačových sítích LAN/WAN prakticky nepoužívá, spíše u sběrnic počítačů. Použití možné i v rádiových sítích, využití vyhrazeného **nízkorychlostního** podkanálu časového multiplexu.
- **Distribuované řízení** – nezávislé na řídící stanici.
  - **Rezervace kanálu** - rezervační rámec, ve svém slotu může každá stanice **požádat o přidělení slotu** v datovém kanále, datové sloty následují podle okamžitého požadovaného počtu za rezervačním rámcem. **Neefektivní pro velký počet stanic** na rozlehlé síti s malou zátěží. Na prázdném kanále neustálé opakování rezervačních slotů.
  - **Binární vyhledávání** - stanice nejprve postupně vysílají bity své adresy od nejvyššího řádu, na sběrnici se logicky sčítají [OR]. Jakmile **stanice vysílá 0 a čte 1**, chce vysílat někdo s vyšší prioritou a stanice musí **zmlknout**.

- **Logický kruh [Token Passing Bus]** - adresy stanic tvoří cyklickou posloupnost, každá stanice zná svou adresu a adresu následníka. Mezi stanicemi se cyklicky předává právo k vysílání (token). Stanice vlastnící token smí vysílat, do určité doby však musí předat token následníkovi. Problém počátečního ustavení posloupnosti, odpojování a připojování stanic do logického kruhu za provozu (rekonfigurace). Velké zpoždění při malé zátěži a velkém počtu stanic.
- **Virtuální logický kruh** - po odvysílání rámce je každý další stanici vyhrazen **časový interval**, kdy smí začít vysílat, nevyužije-li jej, následuje interval další stanice. Nutnost synchronizace stanic. V oblasti malých zátěží efektivnější než logický kruh.

5.2 Kolizní (nedeterministické) metody sdílení média

Prostá Aloha

- Stanice **vysílají bez ohledu na cokoliv**.
- **Časový limit pro potvrzení**, při vypršení náhodné pozdržení (aby nedošlo k opakované kolizi) a opakování pokusu.
- **Kolizní slot**: **2*t0** [t0-doba pro vyslání rámce]. **Dvojnásobek**, protože může být zarušen koncem jiného rámce. Kolizní slot udává, kolik času se nejvýše ztratí na nevyužití kanálu vlivem kolize.

Taktovaná Aloha

- Vysílat se smí začít jen v okamžicích začátků **časových úseků** pro odeslání jednoho rámce.
- **Kolizní slot** je **poloviční** než u prosté Alohy.
- **Exponenciální závislost**, malý vzrůst zátěže může významně zvýšit počet opakování a snížit průchodnost kanálu.

Řízená Aloha

- **Řízená změna intenzity opakování** podle stavu sítě: vyšší intenzita opakování => rychlejší předání rámce, při blížícími se zablokování se intenzita sníží.
- **Případně sledování provozu na kanále** [poměru neobsazených slotů] a nastavování intenzity.

Carrier Sense Multiple Access (CSMA)

Skupina metod **náhodného přístupu s příposlechem nosné**, tj. využití znalosti o obsazení kanálu. Podmínky pro aplikaci: **dokonalá slyšitelnost stanic**, **malé zpoždění signálu** (což platí v LAN).

1. **CSMA (naléhanící CSMA, 1-persistant CSMA)** - před odesláním rámce se testuje stav kanálu, je-li obsazen, odloží se vysílání na okamžik jeho uvolnění. Riziko kolize stanic čekajících na uvolnění kanálu. Při obsazení kanálu čekání náhodnou dobu před dalším pokusem.

2. **Nenaléhající CSMA (non-persistant CSMA)** - při detekci obsazeného kanálu se počká náhodnou dobu pak opět test obsazení. Čekací doba se obvykle volí jako k-násobek doby průchodu signálu sběrnicí.

3. **P-naléhající CSMA (p-persistant CSMA)** - při obsazení kanálu v okamžiku potřeby vysílání se počká na uvolnění kanálu (nebo byl volný okamžitě), pak se **s pravděpodobností p** začne vysílat, s pravděpodobností (1-p) se počká krátký interval, pak se opakuje do úspěšného odeslání rámce. Volbou p lze nastavit optimální využití kanálu pro danou zátěž. Pro p=1 jde o naléhající CSMA.

Carrier Sense Multiple Access with Collision Detection (CSMA/CD)

- CSMA **s detekcí kolize** (sledování vlastního vysílání).
- Využívá se u klasického **Ethernetu**.
- Před vysíláním musí být na médiu klid po dobu kolizního slotu.
- Jinak postup odpovídá naléhající CSMA.
- **Okamžité ukončení vysílání po detekci kolize** - kanál se zbytečně nezaplňuje rámcem, který je stejně zkolidován.
- Závislost maximální délky segmentu na **rychlosti šíření signálu** (vztah s dobou vysílání rámce a minimální délkou rámce)
- Nutnost kódování, které **umožňují detekci kolize** (otevřený kolektor, měření napětí na médiu generovaného proudovými zdroji vysílačů).
- Při **detekci kolize stanice** vysílá kolizní signál **[jam]**, aby kolizi rozpoznaly všechny kolidující stanice. O opakování se stanice pokusí až po náhodné časové prodlevě, stabilitu metody nutno zajistit řízením intenzity opakování.

Carrier Sense Multiple Access with Collision Avoidance (CSMA/CA)

Rezervace intervalu po odvysílání rámce pro potvrzení (v předchozích metodách jsme na potvrzení nahlíželi jako na přídavnou zátěž). Před začátkem vysílání paketu stanice určitý čas poslouchá, zda je přenosové médium volné. Pokud ano, oznámí vysílání a rezervuje přenosové médium a může zahájit vysílání. V opačném případě čeká na konec právě probíhajícího vysílání. Využití v **Wi-Fi**.

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

6.1 Směrování

Směrování (routování) označuje v informatice určování cest datagramů v prostředí počítačových sítí. Směrování zajišťují nejen routery, ale i koncové stanice (při vysílání) a jeho úkolem je doručit datagram (resp. paket) adresátovi, pokud možno co **nejefektivnější cestou**. Směrování zajišťuje **síťová vrstva** modelu ISO/OSI a je využíváno v lokálních sítích LAN i na Internetu, kde jsou dnes směrovány zejména IP packety. Síťová infrastruktura mezi odesílatelem a adresátem paketu může být velmi složitá, a proto se směrování zpravidla nezabývá celou cestou paketu, ale řeší vždy **jen jeden krok**, tj. komu datagram předat jako dalšímu. **Hledání cesty v síti:**

- Ve **spojově orientovaných sítích** při vytváření spojení:
  - Nastavování **spojovacích polí přepínacích prvků** na cestě.
  - Budování **přepínacích tabulek** virtuálního kanálu.
- V sítích s **přepínáním paketů** (obvykle obecně polygonálních a s alternativními cestami) při přenosu každého jednotlivého paketu => každý paket může jít **jinou cestou**.

Směrování v počítačových sítích a v Internetu

Abychom mohli paketovou sítí směrovat pakety od zdroje k cíli, potřebujeme správným způsobem naplnit **směrovací tabulky** všech směrovačů na trase. V malých sítích nebo v sítích, z nichž veškerý provoz ven odchází po jediné implicitní (default) cestě, toto lze vyřešit manuálním vložením potřebných informací, tj. **statickým směrováním**. V rozlehlejších sítích s měnící se topologií (z nichž největší je bezesporu Internet) jsou však nutné **dynamické směrovací protokoly**, které zajistí správné naplnění směrovacích tabulek automaticky na základě výměny informací mezi směrovači.

Hierarchické směrování, autonomní systémy

Současný Internet je natolik **rozsáhlý** a **proměnlivý**, že není reálné udržovat ve směrovačích úplnou informaci o jeho topologii. Tato informace by navíc byla velmi nestabilní, protože by se měnila s výpadkem nebo zapojením linky kdekoli na světě. Proto bylo rozhodnuto směrování v Internetu řešit **hierarchickým způsobem**. Předpokladem jeho použití je rozdělení Internetu do tzv. **autonomních systémů (AS)**. Autonomním systémem rozumíme souvislou skupinu sítí a směrovačů, které jsou pod společnou správou a řídí se společnou směrovací politikou. Pod společnou směrovací politikou si představme zejména dohodnutý vnitřní směrovací protokol (např. OSPF nebo RIP), ale také speciální požadavky administrátorů na směrování některých druhů provozu (traffic engineering, load balancing). Příkladem

autonomního systému tak může být autonomní systém jednoho konkrétního **poskytovatele Internetu (ISP)** nebo velké firmy.

Cílem hierarchického směrování je **vždy nejprve doručit paket** určený pro **některou ze sítí autonomního systému** na hranice tohoto autonomního systému. O další směrování ke konkrétní síti uvnitř AS se již postará vnitřní směrovací protokol, který topologii (nebo alepoň cesty ke všem sítím) svého vlastního AS zná. Směrovač, který je na hranici autonomního systému a účastní se jak na směrování mezi AS tak ve směrovacím protokolu svého AS, se nazývá **hraniční směrovač** (angl. border gateway). V případě neznámé cesty, cesta putuje **default cestou**, která jej pošle nadřazenému AS a ten se o její zpracování dále postará.

Vnitřní a vnější směrovací protokoly

Při směrování v rámci jednotlivých autonomních systémů se používají tzv. **vnitřní směrovací protokoly** - Interior Gateway Protocols, IGP. Naopak pro směrování mezi autonomními systémy se používají **vnější směrovací protokoly** - Exterior Gateway Protocols, EGP. Typickými vnitřními směrovacími protokoly jsou dnes např. **OSPF** nebo starší **RIP**, jako vnější směrovací protokol se používá téměř výhradně protokol **BGP**.

Statické směrování

- **Směrovací tabulky** v jednotlivých směrovačích **konfigurovány "ručně"** - pracnější.
- Odpadá **režie směrovacích protokolů** (zabraná šířka pásma, čas na zpracování).
- **Bezpečnější** (omezení možnosti generování falešných směrovacích informací, odposlouchání topologie sítě).
- Při **výpadku** linky nutný **ruční zásah**.
- Použitelné, pokud se topologie **příliš často nemění** (vlivem výpadků a modifikací sítě).
- V intranetech používáno až překvapivě často.

Dynamické směrování

- **Automaticky reaguje** na poměry v síti (topologie, zátěž, ...).
- Nutnost **provozu směrovacích protokolů**.
- Užitečné **při častých změnách** (příp. i obecně neznámé) topologie sítě (typicky v Internetu).
- V praxi často používána **kombinace statického a dynamického směrování**, staticky nakonfigurované cesty mají obvykle přednost.

Směrovací algoritmy

Vnitřních směrovací protokoly

Rozlišujeme dvě základní třídy směrovacích protokolů: protokoly založené na **vektoru vzdálenosti - DVA** (distance-vector) a na **stavech linek - LSA** (link-state).

DVA - Distance Vector

Sousední směrovače si navzájem vyměňují své směrovací tabulky a doplňují si informace, které se naučí od sousedů, v určitých časových intervalech. **Topologii** celé sítě však **neznají**, musí se spokojit s adresami sousedů, přes která mají posílat pakety do jednotlivých cílových sítí a **vzdálenostmi** k těmto sítím, které společně tvoří tzv. **distanční vektory**.

- Na začátku směrovací tabulka obsahuje **pouze přímo připojené sítě** (staticky nakonfigurováno administrátorem).
- Směrovače poté **periodicky zasílají** směrovací tabulky sousedům.
- Z došlých **směrovacích tabulek sousedů** (vzdáleností sousedů od jednotlivých sítí) si směrovač postupně **upravují a budují** svou směrovací tabulku.
- Pokud cesta **nebyla delší dobu sousedem inzerována**, ze směrovací tabulky se **odstraní**.

Vlastnosti

- **Metrikou** je **počet "přeskoků"** (hop count) na cestě mezi zdrojem a cílem.
  - Nezohledňuje parametry jednotlivých linek (přenosovou rychlost, zpoždění, okamžitou zátěž, ...).
- **Pomalá konvergence při změnách** topologie.
  - O změně se informuje až při příštím periodickém broadcastu směrovací tabulky.
- **Zátěž** od broadcastu celých směrovacích tabulek.
- **Příliš "optimistické"** - směrovač se rychle učí dobré cesty, ale **špatně zapomíná při výpadcích**.
  - **Čekání** na timeout cesty, která přestala být inzerována.
  - Žádný směrovač nikdy nemá metriku horší, než minimum z metrik sousedů + 1 => pomalé šíření špatných zpráv.

Routing information protocol (RIP)

- Velmi starý, stále však **často implementovaný** v malých sítích.

- Hodnota "nekonečna" **řešící problém počítaní do nekonečna** a v technice Poisson reverse reprezentována číslem 16..
- **Maximální počet** přeskoků je omezen na **16**.

Interior Gateway Routing Protocol (IGRP)

- Cisco proprietary.
- **Lepší metrika**, než pouhý počet přeskoků **bandwidth**, **delay**, (**reliability**, **load**).
- **Není omezení** na 16 přeskoků (zvýšeno na 255).

LSA - Link State

- Směrování na základě **znalosti "stavu"jednotlivých linek sítě** (funkčnost, cena).
- Směrovače (uzly grafu) **znají topologii celé sítě** (graf) a **ceny jednotlivých linek** (ohodnocení hran). Tyto informace udržují v topologické databázi. (Všechny směrovače mají stejnou topologickou databázi)
- **Každý směrovač počítá strom nejkratších cest** ke všem ostatním směrovačům (a k nim připojeným sítím) pomocí **Dijkstrova algoritmu**. (Na rozdíl od DVA všechny směrovače počítají na základě **stejných** a **úplných dat**)
- Každý směrovač **neustále sleduje stav** a **funkčnost** k němu připojených linek.
- Testováním linek k sousedním směrovačům – **Hello protokol**.
- Při **změně okamžitě šíří informaci** o aktuálním stavu svého okolí všem ostatním směrovačům. Ty si ji vloží do topologické databáze.
- **Šíří se pouze změny** (ale do celé sítě) - žádné periodické rozesílání směrovacích tabulek.
- **Okamžitá reakce na změnu stavu linek** (výpadek, náběh) => rychlá konvergence
- Zástupci: **OSPF (Open Shortest Path First)**

6.2 Adresování

Adresování sítě přiděluje oblastní správce (pro Evropu RIPE).
IP adresa je **logická adresa** zařízení v počítačové síti (na 3. vrstvě podle OSI modelu), která používá IP protokol. V IP v4 je tato adresa velká 32bitů (4 byte) a zapisuje se pomocí **dot-decimal notation**. To znamená pomocí čtyř dekadických hodnot (každá velká jeden byte), označovaných jako oktet, oddělených tečkou. Příkladem IP adresy je 193.222.5.15. IP adresa se skládá z několika částí. V základu jde o dvě části, **prefix** identifikující síť a **adresa uzlu** v rámci podsítě. Pro to, abychom určily, která část IP adresy je pro **podsíť** a která pro **uzel**, se používá **maska podsítě** (subnet mask). Ta v binárním tvaru obsahuje jedničky následované nulami a pomocí jedniček„vymaskovává" část síťového prefixu v IP adrese. Jinak řečeno, tam kde jsou v masce jedničky je v IP adrese část podsítě a kde jsou

nuly je část uzlu. Síťová část IP adresy určuje podsíť a používá se ke směrování. Část uzlu identifikuje stanici uvnitř daného subnetu.

Maska se v IPv4 zapisuje stejně jako IP adresa pomocí dot-decimal form s tím, že jsou validní pouze adresy, které v binárním zápisu mají zleva jedničky následované nulami (za první nulou smí být pouze nuly). Příkladem síťové masky je 255.255.255.0. Maska 255.255.255.254 není platná, protože uvnitř takového subnetu by se nacházelo 0 uzlů. Maska 255.255.255.255 není maskou podsítě, ale určuje jeden uzel.

Občas se používá speciální zápis, který se označuje jako **inverse mask** nebo **wildcard mask**. Jedná se o obrácenou masku, jednoduše řečeno tam, kde jsou v tradiční masce jedničky jsou nuly a naopak. Pro výpočet v dekadické formě můžeme použít pro každý oktet hodnota = 255 – hodnota oktetu. Například pro masku 255.255.255.240 je inversní maska 0.0.0.15.

Veřejné a privátní adresy

Prvotní princip IP sítí byl takový, že IP adresa musela být unikátní v rámci celé sítě a jednotlivé uzly (stanice, servery a další zařízení s přiřazenou IP adresou) spolu mohli komunikovat. S rozvojem internetu se ukázalo, že počet adres, které je možno v IPv4 vytvořit, rozhodně není dostatečný. Začaly se tedy používat různé metody, jak se s nedostatkem adres vypořádat. Nejrazantnější je nová verze IP protokolu IPv6, která obsahuje mnohem více adres, ale její nasazování není jednoduché. Dále se objevila technika CIDR (Classless Inter-Domain Routing, tj. „beztřídní směrování“) a také se IP adresy rozdělili na dva typy, na veřejné a neveřejné IP adresy.

Veřejné IP adresy (public address) tvoří hlavní část adresního rozsahu internetu a tyto adresy jsou routovatelné v rámci celého internetu. Jednoduše řečeno počítač s veřejnou adresou může být dostupný z celého internetu. Tyto adresy musí být unikátní v celé síti (internetu).

Oproti tomu privátní IP adresy (private adress) by se měly používat pouze v rámci LAN sítí a v internetu by přes ISP neměli komunikovat. Firma pak používá jednu (nebo více) veřejnou adresu a pomocí techniky NAT (Network Address Translation) se při komunikaci mimo LAN překládají privátní adresy na tuto veřejnou. Stejné privátní adresy se tak mohou nacházet na mnoha místech v internetu, ale nemohou spolu přímo komunikovat. Následující tabulka ukazuje jednotlivé privátní rozsahy.

síť	adresa sítě	broadcast adresa	adresy uzlů
10.0.0.0/8	10.0.0.0	10.255.255.255	10.0.0.1 - 10.255.255.254
192.168.0.0/16	192.168.0.0	192.168.255.255	192.168.0.1 - 192.168.255.254
172.16.0.0/12	172.16.0.0	172.31.255.255	172.16.0.1 - 172.31.255.254

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)

6.3 Překlad Adres

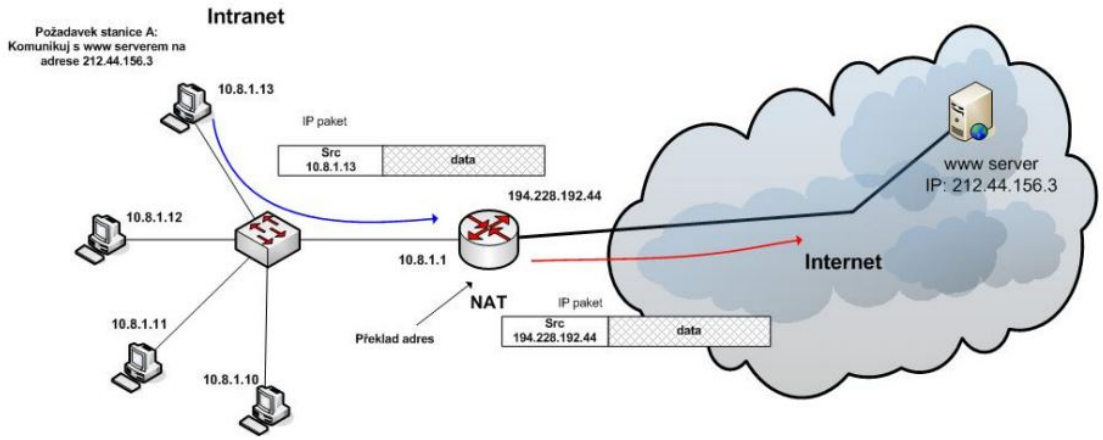
6.3.1 NAT (Network Address Translation)

Překlad adres je důležitou vlastností směrovačů. Používá se zejména ze dvou důvodů:

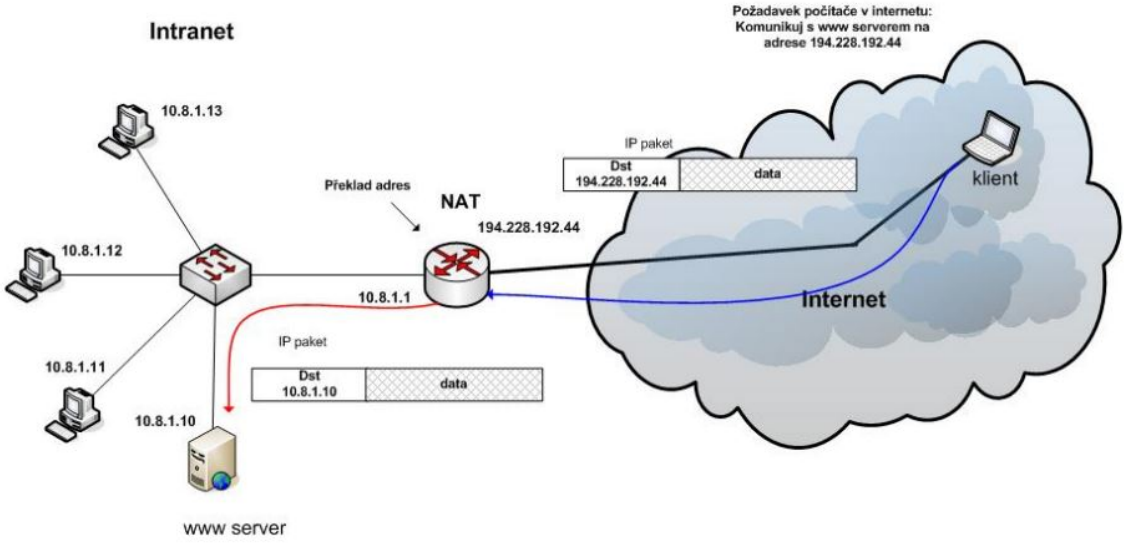
- omezit potřebný počet veřejných IP adres
- zajistit bezpečnost komunikace mezi vnitří a veřejnou sítí

NAT pracuje na 3. (síťové) vrstvě OSI modelu.

Připojení vnitří sítě LAN k internetu pak zajišťuje pouze jediná veřejná IP adresa (*poskytovatel internetu jich může nabídnout více). Ta je přidělena na venkovní rozhraní směrovače, který zajišťuje spojení se sítí internet. NAT může fungovat jak pro příchozí, tak i pro odchozí spojení. U odchozího spojení dochází nahrazení zdrojové adresy z privátního rozsahu veřejnou IP adresou směrovače. Při příchozím spojeních zase dochází ke změně cílové adresy z veřejné na privátní.



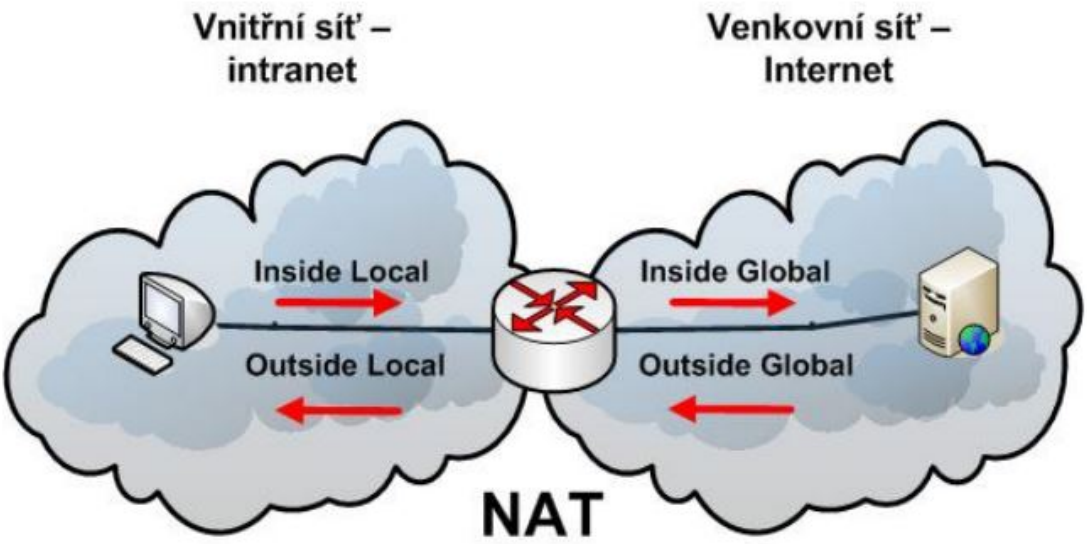
NAT pro odchozí spojení



NAT pro příchozí spojení

Při používání NAT definujeme tyto druhy IP adres:

- **Inside local** – adresa zařízení ve vnitřní síti (počítač), privatní IP adresa
- **Inside global** – veřejná IP adresa, která je viditelná v internetu jako adresa počítače
- **Outside local** – IP adresa z vnitřní sítě, pod kterou je viditelný venkovní počítač
- **Outside global** – veřejná IP adresa vzdáleného serveru v internetu



IP adresy používané při NAT

Technologie NAT pracuje v několika režimech:

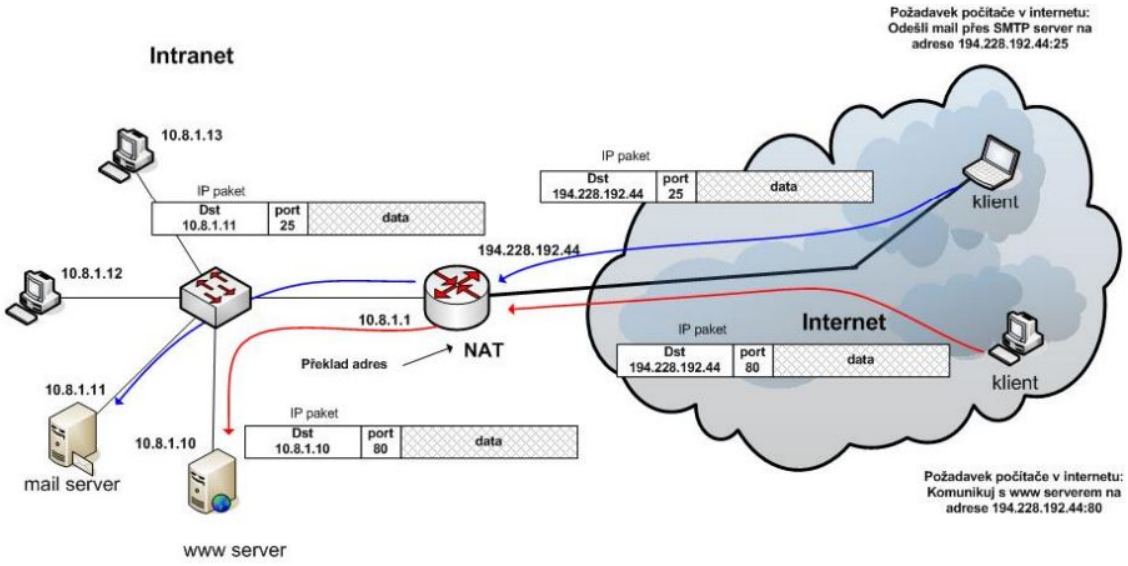
- **Statický NAT** – mapuje vnitřní IP adresu na vnější IP adresu. Je používán pokud je zapotřebí přistupovat z vnější sítě do vnitřní
- **Dynamický NAT** – mapuje vnitřní IP adresu na vnější IP adresu z nějaké skupiny IP adres (adress pool)
- **overloading** – přetěžování mapuje vnitřní IP adresy na jednu vnější IP adresu (bývá označován jako PAT, jedná se o dynamickou formu NATu).

NAT může pro překlad používat jedinou veřejnou IP adresu nebo i více adres. Záleží na množství počítačů ve vnitřní síti a druhu provozu. V případě, že používáme jedinou veřejnou IP adresu mluvíme o přetěžování (overload).

Při používání technologie NAT se setkáváme s různými problémy. Některé aplikace nemusí fungovat správně. To je jedním z důvodů proč ISP (zejména velcí jako UPC, O2, ...) přidělují klientům přímo veřejné IP adresy. Typickým příkladem je protokol IPSec. Jelikož IPSec navazuje komunikaci mezi dvěma koncovými body, e zde zásadním problémem záměna adres v hlavičce paketu.

6.3.2 PAT (Port Address Translation)

V případě PAT dochází k překladu jak adres tak i příslušných portů v IP komunikaci. Výhodou zde je, že za jednu venkovní IP adresu můžeme maskovat celou řadu služeb, které jsou hostovány na různých serverech. Typickým příkladem jsou FTP a www servery umístěné v DMZ.



PAT – maskování dvou serverů za jedinou veřejnou IP adresu

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

- **Utajení** (confidentality) – posluchač na kanále datům nerozumí
- **Autentizace** (authentication) – jistota, že odesílatel je tím, za koho se vydává
- **Integrita** (integrity) – jistota, že data nebyla na cestě zmodifikována
- **Nepopiratelnost** (non-repudiation) – zdroj dat nemůže popřít jejich odeslání

Útoky

- **ARP dotazy** - falšování ARP odpovědí (falešný překlad IP-to-MAC). **Oprava** užitím statických ARP záznamů.
- **Routing protocol** - falšování routovacích informací propagovaných routovacím protokolem (RIP atd). Oprava **filtrováním** zdrojů routovacích informací.
- **Switchované sítě** - proti přetížení přepínací tabulky. Oprava užitím limitování počtu MAC na portu, statickým listem povolených MAC.
- **Brute Force** - zadávání hesel pomocí hrubé síly.
- **Denialenial of service (DoS)** - cílem útočníka vyčerpání systémových prostředků (paměť, CPU, šířka pásma) síťového prvku nebo serveru a jeho zhroucení nebo změna požadovaného chování.
  - **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. **Řešení:** packetové filtry.
  - **SYN flood** – neustálé navazování TCP spojení (příznak SYN), server alokuje prostředky a pošle (SYN-ACK) a čeká na odpověď, které se nedočká. **Řešení:** zkrácení doby čekání na odpověď.
- **Distributed DoS (DDoS)** – DoS útok je vedený z mnoha stanic, které byli již před tím napadeny a nyní jsou využity k tomuto útoku. Je obtížně blokovatelný kvůli přístupu mnoha stanic.

Filtrování provozu

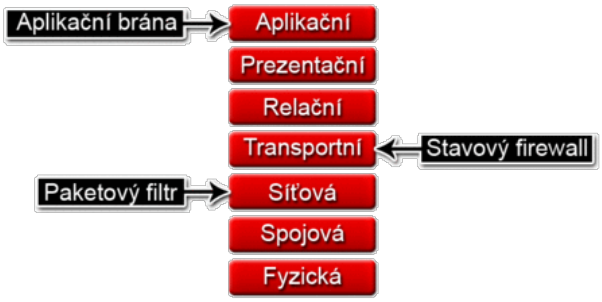
Paketové filtry (nestavové)

Filtrování probíhá dle informací v hlavičce 3 a 4 vrstvy. Pravidla udávají, ze které adresy a portu na kterou adresu a port může být paket procházející rozhraním routeru propuštěn. Na routerech CISCO je realizován jako **Access Control List (ACL)** prostřednictvím sekvence záznamu, které povolují/zakazují přenos paketu, které odpovídají daným

kritériím. Samotný paketový filtr je rychlý, nenáročný na systémové zdroje, ale úroveň kontroly je relativně malá.

Stavový firewall

(též stavový paketový filtr, anglicky stateful firewall) odděluje důvěryhodnou (interní) síť od nedůvěryhodné (externí) sítě. Funguje stejně jako jednoduchý paketový filtr, ukládá **navíc ale i informace o povolených spojeních**, podle kterých pak může rozhodovat, zda procházející pakety patří do již povoleného spojení a mohou být propuštěny nebo zda musí znovu projít kontrolou. Firewall je **velmi rychlý**, poskytuje slušnou úroveň zabezpečení a snazší konfiguraci. Poskytuje urychlené zpracování paketu již povoleného spojení. 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.



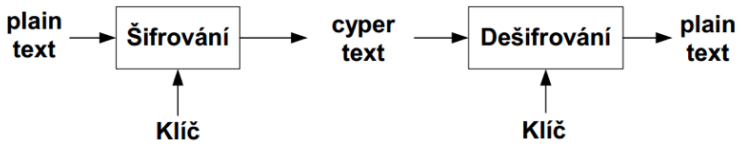
Virtuální privátní síť

(zkratka VPN, anglicky virtual private network) je v informatice prostředek k **propojení několika počítačů prostřednictvím (veřejné) nedůvěryhodné počítačové sítě**. Lze tak snadno dosáhnout stavu, kdy spojené počítače budou mezi sebou moci komunikovat, jako kdyby byly propojeny v rámci jediné uzavřené privátní (a tedy důvěryhodné) sítě. Při navazování spojení je totožnost obou stran ověřována pomocí **digitálních certifikátů**, dojde k autentizaci, veškerá komunikace je šifrována, a proto můžeme takové propojení považovat za bezpečné.

Šifrování

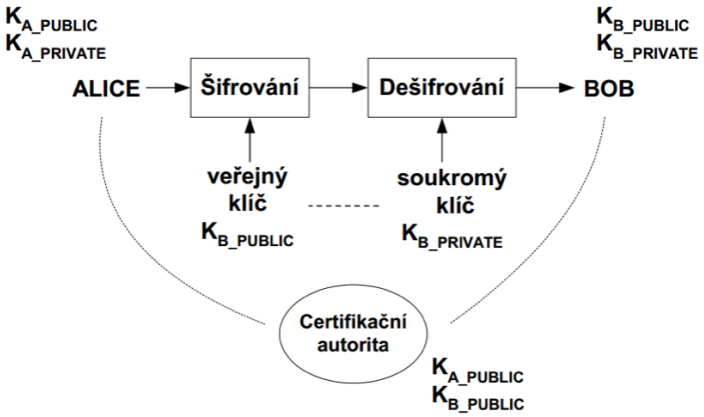
Symetrické šifrování

Pro šifrování i dešifrování se používá **pouze jeden klíč**, který musí mít všichni účastníci, kteří chtějí data šifrovat nebo dešifrovat. **Nebezpečí hrozí při distribuci** tohoto klíče, který pokud bude prozrazen tak všichni účastníci musí začít používat nový klíč. Šifrování je **rychlé** a **jednoduché** pro implementaci (možná implementace přímo v HW). Nejznámější **DES**, **3DES**, **IDEA**. **DES** již dnes není bezpečný, používá se **3DES**, který **DES** zašifruje 3x po sobě.



Asymetrické šifrování

Podstatou je **generace dvou šifrovacích klíčů**, které spolu spolupracují. Tyto klíče se vzájemně matematicky doplňují a je možné oba použít jak pro dešifrování tak šifrování. Veřejný klíč je volně šiřitelný pro všechny, kteří chtějí šifrovat posílaná data. Soukromý klíč je tajný a je pouze pro potřebu pro dešifrování toho, co bylo zašifrováno veřejným klíčem. Uživatel tedy pro šifrovanou komunikaci potřebuje oba klíče. Výhodou je, že **není třeba veřejný klíč speciálně ukrývat** (je vyřešena metoda distribuce), pouze je třeba zajistit mechanizmus proti modifikaci veřejných klíčů při přenosu - **certifikovaná autorita (CA)** klíč digitálně podepisuje. Nevýhodou je, že v porovnání se symetrickým je **pomalejší** z důvodu použitých matematických funkcí. Asymetrické systémy jsou např. **RSA** nebo **DSA**. Stupeň **bezpečnosti se odvíjí od délky** použitého klíče.



Certifikační autorita

- Entita, které je důvěřováno.
- Registruje (podepsané) veřejné klíče.
- První kontakt s certifikační autoritou **musí proběhnout osobně** (získání dvojice podepsaný veřejný-privátní klíč).
- Veřejný klíč certifikační autority musí být **důvěryhodným** způsobem zaveden do každého systému.

Zabezpečení na jednotlivých vrstvách OSI-RM

- **L7** – S/MIME
- **L4** – SSL (jen TCP)
- **L3** – IPSec (nezávislé na médiu i aplikaci)
- **L2** – hop-by-hop, neefektivní

Autentizace

Je proces ověření proklamované identity subjektu. Probíhá nejčastěji jednou ze tří metod:

- Řekneme něco, co známe (**heslo**, PIN)
- Ukážeme něco, co máme (ID karta, **hardwarový klíč**)
- Necháme systémem změřit něco našeho (**biometrické údaje**, otisk prstů, sítnice)

Existuje **několika stupňové ověření**, například kombinací PIN a biometrických údajů. Po ověření identity následuje autorizace což je souhlas k provedení operace či umožnění přístupu. **Často se používá v souladu s šifrováním:**

- U **symetrického** odesílatel šifruje uživatelské jméno sdíleným klíčem a příjemce kontroluje platnost tohoto uživatelského jména.
- U **asymetrického** se užívá digitálních podpisů Certifikační autority.