Cvičení z KIV/UPS -- 2003

Jiří Patera

Motto: "Linux is a train of pain but the tickets are free"

- Toto jsou přípravné texty, podle kterých probíhala cvičení předmětu KIV/UPS na FAV-ZČU v Plzni, která vedl Jiří Patera v roce 2003.

- Jedná se pouze o přípravy, na cvičení mohlo být probíráno i něco navíc nebo něco jiného, zde nepoznamenaného (např. praktické části cvičení, zvídavé dotazy, apod.)

- Autor při přípravě těchto textů využíval různé zdroje. Ty nejpodstatnější jsou uvedeny v textech pod nadpisy příslušných kapitol.

- Texty pravděpodobně obsahují chyby, kterých se autor dopustil při jejich tvorbě. Může za ně z velké části dnešní uspěchaná doba a také automatické opravy programu Micro$oft Word.

- *NO WARRANTY:* Každý, kdo bude používat tyto texty, tak činí na vlastní nebezpečí! Autor nenese žádnou zodpovědnost za škody jimi způsobené ;o)

Unless otherwise stated, the text in this PDF file is provided "as-is," without any express or implied warranty. In no event shall the author be held liable for any damages arising from the use of this text.

"1337-5p34k" "! h0p3 7h!5 73x7 w!11 h31p y00, d00d" "533 y00 4r0uNd, 630r63"

1. cvičení z UPS

Úvod, základní informace, podmínky získání zápočtu.

Cvičící: Ing. Jiří Patera
Konzultační hodiny: Po 9.00-10.00, Út 11.00-12.00
Kancelář: UL408
E-mail: jipatera@kiv.zcu.cz
Homepage: http://zcu.cz/~jipatera

Oficiální stránka předmětu: http://www.kiv.zcu.cz/~simekm/vyuka/ups

Stránka předmětu u cvičícího: http://zcu.cz/~jipatera/vyuka/ups

- Požadavky na zápočet
  - Semestrální práce
    - Programovací jazyk C, C++, Java
    - OS Linux
  - Dokumentace
    - Papírová
    - WWW stránka
  - Plán cvičení
  - Mezní termíny (*deadlines*)
- Diskusní fórum

2. cvičení z UPS

Operační systém UNIX (Linux), příkazy.

http://www.kiv.zcu.cz/~simekm/skoleni/index.html
http://earchiv.isdn.cz/ascii/i_coje.php3

Základy počítačové sítě – referenční model ISO/OSI

- **Fyzická úroveň:** Zajišťuje přenos jednotlivých bitů od odesilatele k příjemci.
    - Patří sem kabely (UTP, STP, koaxiály, světlovody). Na této vrstvě ISO-OSI pracují zařízení jako transeivery (přizpůsobení signálu médiu) nebo opakovače (zesilovače signálu, zejména pro dlouhá vedení), ale částečně i síťové karty.
- **Linková úroveň:** Určuje, co jednotlivé bity přenesené fyzickou vrstvou znamenají (zda se jedná o řízení nebo data).
    - Datovou jednotkou (skupina bitů) linkové vrstvy je **rámec**.
    - Přenáší data mezi dvěma uzly, které mají přímé spojení.
    - Mezi zařízení pracující na linkové úrovni patří síťové karty, mosty a přepínače. Tato zařízení mají společné to, že pracují s tzv. MAC adresou (fyzickou nebo HW adresou), která je tvořena 6-ti bajty (zápis 01:02:03:04:05:06). Tato adresa je dána výrobcem a je pro každou NIC unikátní na celém světě, tzv. flat address scheme.
- **Síťová úroveň:** Zajišťuje komunikaci dvou uzlů počítačové sítě, mezi kterými neexistuje přímé spojení, je tudíž nutné pro ně najít alespoň spojení nepřímé, kterých může existovat i více.
    - Datovou jednotkou síťové vrstvy je **paket**.
    - Mezi zařízení pracující na síťové úrovni patří např. směrovače. Tato zařízení mají společné to, že pracují s tzv. IP adresou (síťovou nebo internetovou adresou), která je tvořena 4-mi bajty (zápis tzv. tečková notace 147.228.63.47). Tato adresa je přidělována jedinou organizací a je tudíž také unikátní na celém světě, tzv. hierarchické adresovací schema.
    - Třídy IP adres: A (0), B (10), C (110), D (1110), E (1111). Speciální IP adresy (.0, .255, případně .1). Zmínka o DNS (root servery, top domény, převod IP na jméno pomocí 47.63.228.147.in-addr.arpa).
    - Síťová maska: 255.255.255.0 (třídy C), slouží jako možnost vytvářet lokální podsítě v přiděleném síťovém rozsahu. Bitovým logickým součinem získáme adresu sítě.
- **Transportní vrstva:** Poskytuje vyšším vrstvám (aplikacím) kvalitnější přenosové služby, než jaké ve skutečnosti dokáže poskytovat vrstva síťová.

Umožňuje vytvářet spojení spojovaného (TCP) i nespojovaného (UDP) charakteru.

  - Datovou jednotkou transportní vrstvy je **segment**.
  - Mezi zařízení pracující na této úrovni patří např. transportní brány (filtrují data v závislosti na hodnotě zdrojového/cílového portu).
  - V této vrstvě se adresuje cílová aplikace pomocí tzv. portu, což je 16-bitové unikátní číslo na daném počítači (0-1024 systém, 1024+ aplikace).

- **Relační vrstva:** Poskytuje vyšším vrstvám o něco více služeb než nižší vrstvy.

  - Relace: jako telefonní hovor, kde se mění lidé u aparátu, aniž by byl zavěšen.
  - Při řízení dialogu se využívá TWS (full-duplex), TWA (half-duplex, pověření), One-Way (simplex). Synchronizace se provádí pomocí hlavních a vedlejších synchronizačních bodů.

- **Prezentační vrstva:** Nižší vrstevy ISO/OSI zajišťují, že přenášená data dorazí k příjemci v podobě, v jaké byla vyslána. Stejná "podoba" však ještě nezaručuje, že pro příjemce nebudou jedna a tatáž data představovat něco jiného, než pro jejich odesilatele.

  - Proto se využívá ASN.1 (odesilatel zakóduje data pomocí ASN.1 a příjemce je dekoduje).
  - Na této úrovni lze také realizovat kompresi nebo kódování dat.

- **Aplikační vrstva:** Poskytuje aplikacím přístup do prostředí sítě na bázi ISO/OSI modelu.

  - Aplikační entita: Část programového vybavení pracující přímo se sítí (nemusí nutně být součástí každé aplikace).
  - Aplikační entita se skládá z prvků ASE (Application Service Elements)
  - ASE se dělí na CASE (Common Application Service Element -- zajišťují služby potřebné pro podporu aplikací různých typů) a SASE (Specific Application Service Element -- realizují specifické služby, potřebné jen pro jeden konkrétní typ aplikací).
  - CASE: Asociace je logické spojení na úrovni aplikační vrstvy. O správu asociací se stará prvek ACSE (Association Control Service Element). O správu RPC se pak stará ROSE (Remote Operations Service Element).
  - Protokoly VT (telnet), FTAM (FTP), CMIP (SNMP)

Základy počítačové sítě – model TCP/IP

- Vrstva síťového rozhraní: odpovídá sjednocení fyzické a linkové vrstvy ISO/OSI modelu.
- Síťová vrstva: odpovídá síťové vrstvě ISO/OSI modelu.

- Transportní vrstva: odpovídá transportní vrsvě ISO/OSI modelu.
- Aplikační vrstva: odpovídá sjednocení relační, prezentační a aplikační vrstvy ISO/OSI.

Souborový systém UNIXu (Linuxu)

- Pod Unixem máme jen jeden hlavní (kořenový) adresář. Každý oddíl (disk) ve vašem systému je pouze mountován (připojován) do nějakého podadresáře v onom hlavním adresáři. Kořenový adresář značíme pouze /.

- Podadresáře kořenového adresáře
    - `/bin` – základní spustitelné soubory (shell a jeho příkazy)
    - `/dev` – soubory všech dostupných zařízení (se vším se v linuxu zachází jako se soubory, tj. porty, disky, periferní zařízení)
    - `/etc` – veškeré konfigurační soubory systému (FTPD, HTTPD, SSHD, startovací skripty systému, správa uživatelů – tj. uživatelé a hesla, skupiny)
    - `/home` – domovské adresáře uživatelů (víceuživatelský OS)
    - `/proc` – virtuální souborový systém obsahující informace jádra (zavedené moduly, HW informace)
    - `/root` – domovský adresář superuživatele
    - `/sbin` – spustitelné soubory, které jsou spouštěny při startu systému nebo superuživatelem (nejsou dostupné běžný, uživatelům)
    - `/lib` – knihovny systému (např. `libc` – knihovna jazyka C)
    - `/usr` – nejrozsáhlejší adresář systému, patří sem veškeré zdrojové kódy systému, všechny používané programy, systémová dokumentace, atd.
    - `/var` – logovací soubory a další obecně často se měnící data
    - `/tmp` – dočasné soubory, každý může přistupovat r/w

- Základní příkazy pro práci se souborovým systémem

    - Práce se soubory: `pwd`, `cd`, `mkdir`, `rmdir`, `ls`, `cp`, `mv`, `rm`
    - Práce s daty souborů: `cat`, `more`, `less`, `head`, `tail`
    - Vyhledávání souborů: `which` (program v `PATH`), `whereis` (soubor v `PATH`, manuálové stránky, zdrojové a konfigurační soubory), `find` (hledá cokoliv a kdekoliv, syntaxe `find dir -name jmeno`), `locate` (prohledává aktualizovanou databázi celého disku, databázi lze aktualizovat i ručně pomocí `updatedb`)
    - Komprese souborů a adresářů: `gzip` soubor (zabalí `.gz` a vymaže soubor), `gunzip` (rozbalí soubor), `tar` (zabalí/rozbalí více souborů příp. adresářů do/z `.tar.gz`, parametry `-xzvf` archiv pro rozbalení, `-czvf` archiv zdroj pro zabalení), `zip -r archiv zdroj` (rekurzivní zabalení), `unzip archiv` (rozbalení).

- Vlastnictví a přístupová práva souborů a adresářů

    - vlastnictví soub. a adresářů (vlastník, skupina, změna `chown`, `chgrp`)

- přístupová práva (4 osmičkové cifry, `xUGO`, `rwx`, měnit lze pomocí `chmod`)
- odkazy na soubory (pevné a symbolické – `ln soubor odkaz`, `ls -s soubor odkaz`), převážně se používají symbolické odkazy
- mountování (`/etc/fstab`, `mount -t fstype device dir`)

- Dokumentace: manuálové a info stránky
  - každý příkaz obsahuje stručnou nápovědu, kterou lze vyvolat buď pomocí přepínače `-h` nebo `--help`
  - v systému je téměř všechno popsáno pomocí manuálových stránek, které lze vyvolat příkazem `man keyword` nebo `man sekce keyword`
  - Sekce 1-8: uživatelské příkazy, systémová volání, knihovní funkce, speciální soubory, formáty souborů, hry, různé, administrace a privilegované příkazy
  - info stránky jsou modernější verze manuálových stránek (umožňují odkazy), vyvolávají se stejně jako manuálové stránky (jen klíčové slovo `info` namísto `man`)

- Základy přesměrovávání v Stellu
  - speciální znaky: `?`, `*`, `\`, `;`, `>`, `>>`, `<`, `|`, `&`, `$`
  - přesměrování chybového výstupu `2>` (do souboru), `2>&1` (na standardní výstup)
  - roury: `grep`, `more`
  - práce s úlohami: `&`, stisk `ctrl+z` (pozastaví úlohu a dá ji na pozadí), `bg` (zpoždění na pozadí pozastavené úlohy), `fg` (přenesení úlohy na popředí), `jobs` (seznam úloh)
  - práce s procesy: `ps`, `ps -aux`, PID, `kill`, `kill -9`, `killall`, `top`

- Základy editoru `vi`
  - šipky, `i` (mód vkládání), `R` (mód přepisování), `:q!` (ukončení bez uložení změn), `:e filename` (otevření souboru `filename`), `:wq!` (ukončení s uložením změn), `:w filename` (uložení do souboru `filename`)

- Přihlášení na vzdálený stroj: `telnet`, `ssh`, `ftp`, `sftp` – parametrem je URL (`user:passwd@hostname`)

- Skriptování:
```
#!/bin/sh
echo "Hello, world!"
```

- Znalost regulárních výrazů vám dá do rukou mimořádně silný nástroj pro práci s textem:
  - vytahovat z textových dat údaje, které vás zajímají,
  - přetvářet je do podoby, kterou potřebujete,
  - vyhledávat a nahrazovat v textových editorech a dalších programech.

3. cvičení z UPS

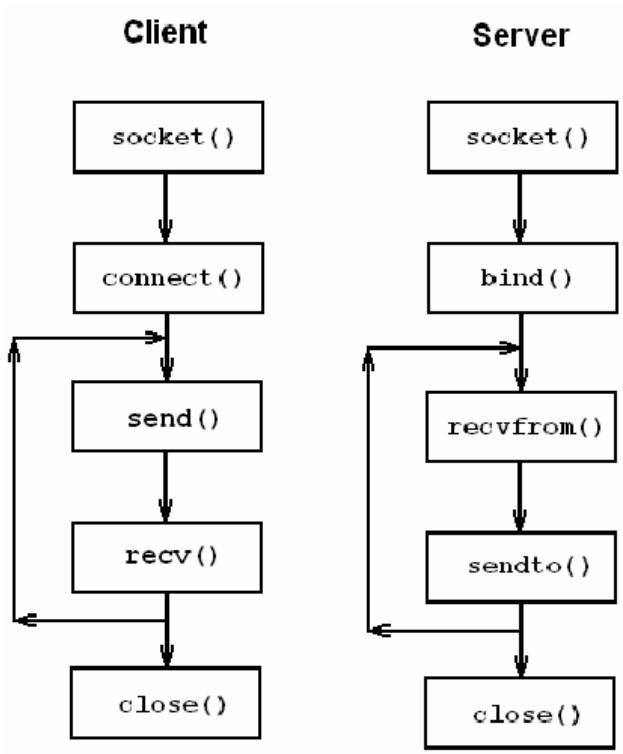
Operační systém UNIX (Linux), BSD sokety.

http://www.kiv.zcu.cz/~simekm/vyuka/materialy/bsd.ps
http://www.kiv.zcu.cz/~simekm/vyuka/materialy/sock.ps

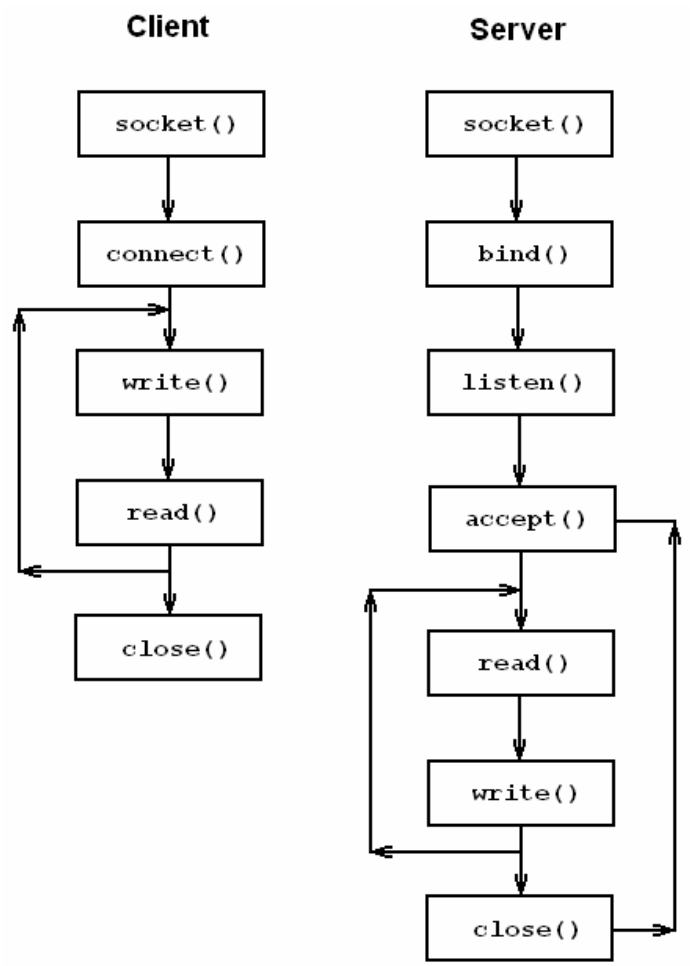
Základní funkce pro práci se sokety

- **Soket:**
  - Datová abstrakce pro komunikaci s koncovým uzlem.
  - Dvojice: IP adresa, port. Tato dvojice jednoznačně definuje nějaký program v Internetu (konkrétní počítač a program).

- **Datagramové služby:** Nespolehlivé, nespojované.
  - `socket(SOCK_DGRAM)` – vytvoření datové struktury soketu
  - `connect()` – připojení ke vzdálenému soketu
  - `bind()` – připojení k lokálnímu soketu
  - `close()` – zrušení (dealokace) datové struktury soketu
  - `send()` – odešle datagram
  - `sendto()` – odešle datagram specifikovanému příjemci
  - `recv()` – přečte datagram
  - `recvfrom()` – přečte datagram a zjistí odesilatele



- **TCP služby:** Spolehlivé, spojované.
    - `socket(SOCK_STREAM)` – vytvoření datové struktury soketu
    - `connect()` – připojení ke vzdálenému soketu
    - `bind()` – připojení k lokálnímu soketu
    - `listen()` – určuje délku vstupní fronty soketu, používá pasivní soket
    - `accept()` – přijme volání klienta, vytvoří pro něj vzdálený soket
    - `close()` – zrušení (dealokace) datové struktury soketu
    - `read()` – čtení dat ze soketu
    - `close()` – zápis dat do soketu



- **Nastavení vlastností soketu:** Nastavení parametrů soketu (timeout, buffer, promiskuitní režim, atd.)
    - `setsockopt()`
    - `getsockopt()`

- **DNS jména a IP adresy:**

    - `gethostname()` – vrátí doménové jméno lokálního uzlu
    - `gethostbyname()` – převod doménového jména uzlu na IP adresu
    - `gethostbyaddr()` – převod IP adresy na doménové jméno uzlu

- **IP adresy a jejich síťová reprezentace:**

    - `inet_ntoa()` – převod 32-bit čísla na IP adresu v tečkové notaci
    - `inet_aton()` – převod IP adresy na 32-bit číslo

- **Lokální a síťová číselná reprezentace:**

        - Big-Endian vs. Little-Endian
        - Network vs. Host Byte Order

    - `ntohs()` – převod ze síťové reprezentace na lokální
    - `htons()` – převod z lokální reprezentace na síťovou

Základní funkce balíčkovacího systému .DEB a .RPM

- **Debian linux:** program `dpkg`

    - Instalace balíčku: `dpkg -i package_name`
    - Deinstalace balíčku (ponechání cfg): `dpkg -r package_name`
    - Kompletní deinstalace balíčku: `dpkg -P package_name`
    - Výpis nainstalovaných balíčků: `dpkg -l`

- **RedHat linux:** program `rpm`

    - Instalace balíčku: `rpm -i package_name`
    - Deinstalace balíčku: `rpm -e package_name`
    - Výpis nainstalovaných balíčků: `rpm -qa`

4. cvičení z UPS

Důležité struktury pro tvorbu semestrální práce. Systém X-Window, psaní vlastních programů pod Linuxem, Makefile.

http://www.kiv.zcu.cz/~simekm/skoleni/index.html

- Tyto struktury lze nalézt v hlavičkových souborech v adresáři `/usr/include`.
  - Struktura `servent` (`netdb.h`):

```
/* Description of data base entry for a single service.  */
struct servent
{
  char *s_name;                 /* Official service name.  */
  char **s_aliases;             /* Alias list.  */
  int s_port;                   /* Port number.  */
  char *s_proto;                /* Protocol to use.  */
};
```

  - Struktura `hostent` (`netdb.h`):

```
struct hostent
{
  char *h_name;                 /* Official name of host.  */
  char **h_aliases;             /* Alias list.  */
  int h_addrtype;               /* Host address type.  */
  int h_length;                 /* Length of address.  */
  char **h_addr_list;           /* List of addresses from name server.  */
#define h_addr  h_addr_list[0]  /* Address, for backward compatibility.  */
};
```

  - Struktura `protoent` (`netdb.h`):

```
struct protoent
{
  char *p_name;                 /* Official protocol name.  */
  char **p_aliases;             /* Alias list.  */
  int p_proto;                  /* Protocol number.  */
};
```

  - Struktura `sockaddr_in` (`netinet/in.h`):

```
/* Structure describing an Internet socket address.  */
struct sockaddr_in
  {
    __SOCKADDR_COMMON (sin_);
    in_port_t sin_port;                 /* Port number.  */
    struct in_addr sin_addr;            /* Internet address.  */

    /* Pad to size of `struct sockaddr'.  */
    ...
  };
```

- Struktura `in_addr` (`netinet/in.h`):

```
/* Internet address.  */
typedef uint32_t in_addr_t;
struct in_addr
  {
    in_addr_t s_addr;
  };
```

- Struktura `iphdr` (`netinet/ip.h`):

```
struct iphdr
  {
#if __BYTE_ORDER == __LITTLE_ENDIAN
    unsigned int ihl:4;
    unsigned int version:4;
#elif __BYTE_ORDER == __BIG_ENDIAN
    unsigned int version:4;
    unsigned int ihl:4;
#else
# error "Please fix <bits/endian.h>"
#endif
    u_int8_t tos;
    u_int16_t tot_len;
    u_int16_t id;
    u_int16_t frag_off;
    u_int8_t ttl;
    u_int8_t protocol;
    u_int16_t check;
    u_int32_t saddr;
    u_int32_t daddr;
    /*The options start here. */
  };
```

- Struktura `udphdr` (`netinet/udp.h`):

```
struct udphdr {
  u_int16_t     source;
  u_int16_t     dest;
  u_int16_t     len;
  u_int16_t     check;
};
```

- Struktura `tcphdr` (`netinet/tcp.h`):

```
struct tcphdr
  {
    u_int16_t source;
    u_int16_t dest;
    u_int32_t seq;
    u_int32_t ack_seq;
#  if __BYTE_ORDER == __LITTLE_ENDIAN
    u_int16_t res1:4;
    u_int16_t doff:4;
    u_int16_t fin:1;
    u_int16_t syn:1;
    u_int16_t rst:1;
    u_int16_t psh:1;
    u_int16_t ack:1;
    u_int16_t urg:1;
    u_int16_t res2:2;
#  elif __BYTE_ORDER == __BIG_ENDIAN
    u_int16_t doff:4;
    u_int16_t res1:4;
    u_int16_t res2:2;
```

```
    u_int16_t urg:1;
    u_int16_t ack:1;
    u_int16_t psh:1;
    u_int16_t rst:1;
    u_int16_t syn:1;
    u_int16_t fin:1;
#  else
#   error "Adjust your <bits/endian.h> defines"
#  endif
    u_int16_t window;
    u_int16_t check;
    u_int16_t urg_ptr;
};
```

- **Systém X-Windows:**
  - X Window Systém (také X-Window nebo jenom X) vznikl na MIT a lze jej definovat jako implementaci GUI. Současná verze je X11 a stále se inovuje. V současné době se nejvíce používá X11 Release6.
  - X Window bylo vyvinuto jako samostatný program (či soubor programů) a není tedy součástí jádra, jak je tomu zvykem u jiných GUI.
    - systém je nezávislý na OS, na kterém byl implementován
    - je možné jej používat distribuovaně (v počítačové síti), tzn. že aplikace mohou běžet na jiném počítači, než na kterém se zobrazují
  - Důležité pojmy:
    - Displej: technické zařízení, na kterém aplikace zobrazuje své okno a z jehož klávesnice čte příkazy a jehož myší ovládá kurzor.
    - Obrazovka: jeden displej může mít více obrazovek.
    - X server: program, který řídí činnost displeje.
    - X client: aplikace, která je někde spuštěna a na nějakém displeji zobrazuje okno.
    - X terminál: počítač bez disku s rychlým CPU, RAM pamětí, klávesnicí, myší a zpravidla velkou obrazovku. V procesoru takového zařízení běží pouze X-server.
  - Displej a obrazovka, na které se má X-klient zobrazovat se určuje následující volbou:
    - `-display [počítač]:displej[.obrazovka]`
    - `[počítač]` je adresa počítače, na kterém běží X server. Pokud adresu neuvedeme, zobrazí se okno na tom počítači, na kterém běží X klient.
    - `[displej]` je číslo X serveru, který spravuje požadovaný displej. Displeje se číslují od 0 a právě hodnota 0 se na tomto místě nejčastěji používá.
    - `[obrazovka]` je číslo obrazovky v rámci zadaného displeje. Nejčastější a implicitní hodnota je 0.

- Pokud při startu X-klienta neuvedeme volbu `-display`, bude se hledat proměnná prostředí `DISPLAY`. Existuje-li, použije se řetězec s ní spojený tak, jako by byl uveden za `-display`.
- X Window disponuje ochranou, která dovoluje zobrazovat okna pouze na tom displeji, jehož uživatel to explicitně povolil příkazem `xhost` (sdělí lokálnímu X-serveru adresy počítačů, ze kterých mohou X-klienti displej používat.
- Musíme se normálně přihlásit na vzdálený počítač např. pomocí programu `ssh`. Potom spustit X-klienta, kterému je ovšem nutné oznámit, se kterým X-serverem má spolupracovat. To je možné provést nastavením proměnné prostředí `DISPLAY`.

♦ **Příkaz `make` a soubor `makefile`:**

- Program `make`:
  - Programy se většinou skládají z několika odděleně překládaných modulů. A právě v těchto případech bývá netriviální záležitostí udržet pořádek a přehled v tom, které moduly mají či nemají být znovu překládány.
  - Situace ale může být ještě komplikovanější, pokud některé soubory závisí na souborech jiných. Často se toho využívá. Typicky jsou to `.c` soubory, které závisí na svých `.h` souborech.
  - Řešením samozřejmě je vše přeložit a sestavit pomocí nějakého scriptu. Script je v UNIXu něco jako dávkový soubor v DOSu. Takovéto řešení není ovšem vůbec elegantní a překládat či sestavovat nezměněné soubory je zcela zbytečné.
  - Pro tyto případy poskytuje UNIX utilitu `make`, která dokáže řídit posloupnost akcí v závislosti na mnoha okolnostech. Navíc např. nepřekládá již přeložené části programů.
  - Definice: Program `make` je nástroj pro správu projektů, vhodný zejména pro programování rozsáhlejších úloh. Udržuje, aktualizuje a obnovuje skupinu programů.
  - Syntaxe: `make [ -f makefile ]`.
- Soubor `makefile`:
  - Parametr `makefile` označuje soubor, který obsahuje závislosti souborů a popisy cílů. Implicitní jména jsou: `makefile`, `Makefile`, `s.makefile`, `s.Makefile`.
  - Znak `#` uvozuje komentář. Nikdy nenahrazujte tabulator mezerami jinak se `make` v `makefile` nevyzná! Ukázka `makefile`:

```
CFLAGS= -Wall
GCC= gcc

all: udp-client udp-server

udp-client: udp-client.o
	$(GCC) $(CFLAGS) -o udp-client udp-client.o
udp-client.o: udp-client.c
	$(GCC) $(CFLAGS) -c udp-client.c

udp-server: udp-server.o
	$(GCC) $(CFLAGS) -o udp-server udp-server.o
udp-server.o: udp-server.c
	$(GCC) $(CFLAGS) -c udp-server.c

clean:
	rm -f udp-client.o udp-client
	rm -f udp-server.o udp-server
```

- Vstupy ve tvaru `string1=string2` jsou definicí makra.
- Výskyty `$(string1)` jsou pak nahrazeny obsahem `string2`.

5. cvičení z UPS

Rozhraní BSD socketů, různé realizace serverů, zadání zápočtových úloh.

http://www.kiv.zcu.cz/~simekm/vyuka/ups/cviceni/priklady.html

Příklady TCP/UDP nad Berkeley sockety

- Obecně:
  - Příklady implementují echo službu (servery opakují zpět klientům co dostanou, klienti vyšlou data na server a zase je přijmou). Klienti jsou `udp/udp.c` a `tcp/tcp.c`, servery jsou `*/udps.c` a `*/tcps.c`.
- Adresace:
  - Příklad využití jmených služeb je v `udp/udp.c`, v ostatních příkladech je použití vynecháno pro větší přehlednost. Typy servrů:
- UDP:
  - Jednoduchý server (`udp.c/udps.c`) s postupným zpracováním po jednom dotazu.
  - Server s předspuštěnými připravenými podprocesy (`udp/ser-pre/udps.c`). Všimněte si uklízeni procesů (`reaper()`, `ender()`) a při běhu zpracování jednotlivými klienty.
- TCP:
  - Jednoduchý server (`tcp.c/tcps.c`) s postupným zpracováním po jednom dotazu.
  - Server s předspuštěnými připravenými podprocesy (`tcp/ser-pre/tcps.c`). Všimněte si uklízení procesů (`reaper()`, `ender()`) a při běhu zpracování jednotlivými klienty.
  - Server se spouštěnými podprocesy na základě požadavku na spojení (`tcp/on-dem/tcps.c`).

- Server se spouštěním vláken na základě požadavku na spojení (`tcp/on-dem-thead/tcps.c`).

- Server (`tcp/other/tcps.c`), který spouští programy pro zpracování požadavků na spojení (`tcp/other/tcps-child.c`). Všimněte si předávání spojení na "well-known" socketu/file descriptoru 5. (v praxi se spíše využívá `stdin`/`stdout` tj. 1/2 (`inetd` -> `telnetd`, `ftpd`, ...)).

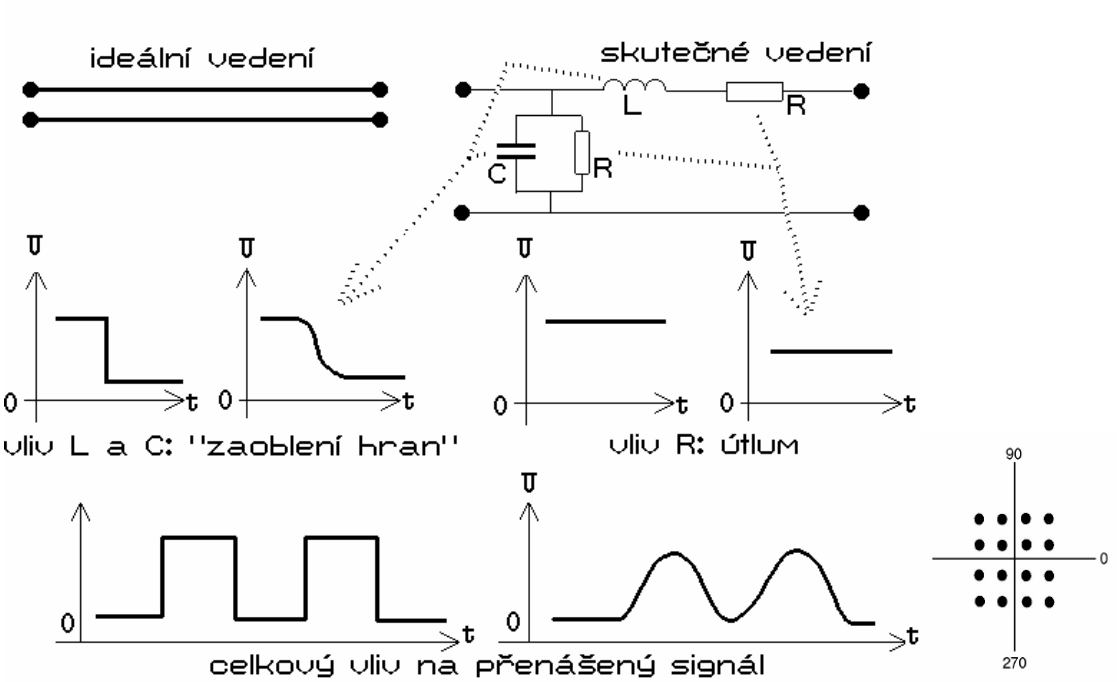
- Server (`tcp/single/tcps.c`), který zpracovává všechny požadavky paralelně. Paralelizmus v rámci procesu je řešen použitím funkce `select()`.

Příklady je možné získat i v komprimované podobě.

6. cvičení z UPS

Kódování a přenos signálu.

- Přenos signálu ideálním a skutečným vedením.



- Typy modulací: AM, FM, PM, QAM-16

- **Nyquist: Modulační rychlost** (neboli rychlost, s jakou dochází k přechodům analogového signálu mezi stavy, reprezentujícími jednotlivé diskrétní hodnoty), může být maximálně rovna dvojnásobku šířky přenosového pásma.

  $\mathbf{f_m \leq 2\,W}$ W … šířka přenosového pásma

- Má-li signál pouze dva stavy (1, 0), pak je jeho přenosová rychlost rovna modulační rychlosti. Pokud má stavů více, pak je i přenosová rychlost vyšší.

- Shannon: **$C = W \log_2(1 + \text{odstup signál/šum})$** [bit/sec.]
  odstup signál/šum = 10 log ($P_{signál}$ / $P_{šum}$) [dB]

  C … kapacita kanálu (max. přenosová rychlost), W … šířka pásma

- Nyquist: **$C = 2\,W \log_2 V$** [bit/sec.]

  C … kapacita kanálu, W … šířka přenáš. pásma, V … počet úrovní signálu

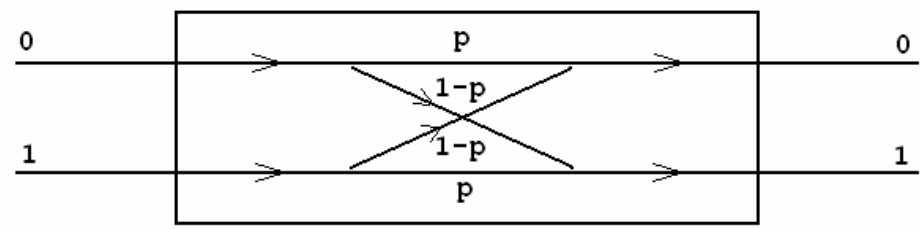
- Typy přenosů
    - Asynchronní – mezi příjemcem a vysílajícím neexistuje žádná synchronizace, speciální značky, přenos jednoho bitu může trvat libovolně dlouhou dobu
    - Arytmický – mezi příjemcem a vysílajícím existuje synchronizace na začátku a na konci přenosu bloku bitů, START/STOP bity, délka přenosu znaku je pevná, délka přenosu bloku proměnlivá
    - Synchronní – mezi vysílajícím a přijímajícím existuje synchronizace po celou dobu, hodiny jsou zakódovány do přenášených dat; NRZ, diferenciální manchester, …

- Synchronizace
    - Bitová – synchronizace na úrovni bitů (rozlišování jednotlivých bitů)
        - Speciální oddělovací značky (hodně režie)
        - START/STOP bity
    - Bytová (znaková) – synchronizace na úrovni bytů (znaků), určování hranic jednotlivých bytů (znaků)
        - START-STOP bity
    - Rámcová – synchronizace na úrovni rámců, určování hranic jednotlivých rámců

- Přenos v základním a rozšířeném pásmu
    - BASEband – přenáší se hodnoty napětí (digitální signál); unipolární a bipolární kódování, manchester, …
    - BROADband – přenáší se signál nejvíce vyhovující přenosovému médiu (nejčastěji harmonický signál); AM, FM, PM

- Časový a frekvenční multiplex

- Chybovost symetrického binárního přenosového kanálu bez paměti
    - Symetrický – 1 nebo 0 se přenáší se stejnou pravděpodobností
    - Binární – přenáší se 1 nebo 0
    - Bez paměti – nezáleží na tom, co se přeneslo v předešlém kroku

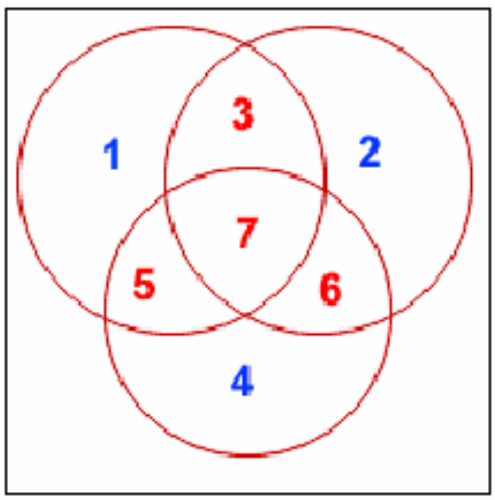


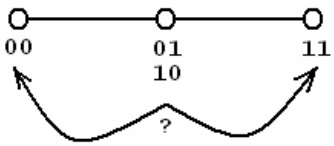
- Pravděpodobnost bezchybného přenosu jednoho bitu je $P_1=p$.
- Pravděpodobnost bezchybného přenosu N bitů bitu je $P_N=p^N$.
- Př.: Mějme dán symetrický binární přenosový kanál bez paměti. Chceme zjistit kolik bitů můžeme přenést (N), aby pravděpodobnost jejich bezchybného přenosu byla $p_N = 0.9$.

  N=?, $p_N = 0.9 \Rightarrow 0.9 = p^N \Rightarrow \ln(0.9) = N \ln(p) \Rightarrow N = \ln(0.9)/\ln(p)$

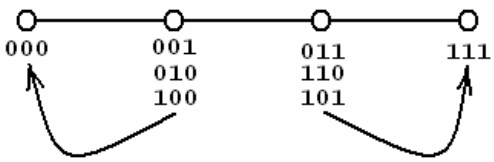
- Hamming's Code
  - Ze 7mi přepravovaných bitů jsou 4 datové a 3 paritní
  - Parita (parita se používá sudá)
    - Sudá (*even*) – paritní bit je volen tak, aby data + parita mělo sudý počet jedniček
    - lichá (*odd*) – paritní bit je volen tak, aby data + parita mělo lichý počet jedniček
  - Hammingova vzdálenost
  - $d_{min} \geq 2\,t$ … detekce t-bitových chyb, oprava (t–1)-bitových chyb

7	6	5	4	3	2	1	
D	D	D	P	D	P	P	7-BIT CODEWORD
D	-	D	-	D	-	P	(EVEN PARITY)
D	D	-	-	D	P	-	(EVEN PARITY)
D	D	D	P	-	-	-	(EVEN PARITY)

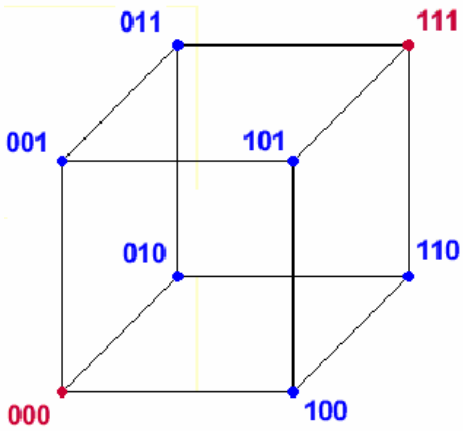




Minimální Hammingova vzdálenost kódu je 2.
Jednobitová chyba jde detekovat, ale nelze opravit.



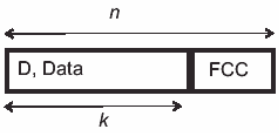
Minimální Hammingova vzdálenost kódu je 3.
Jedno a dvoubitová chyba jdou detekovat.
Opravit lze pouze jednobitovou chybu.



- CRC (Cyclic Redundancy Check)

CRC pomocí aritmetiky mod 2

- pomocí operace eXclusive-OR (XOR)
- popis parametrů
  T = n-bitový rámec, který se přenáší
  D = k-bitový blok dat, prvních k bitů T
  F = $(n–k)$-bitová FCS; posledních $(n–k)$ bitů T
  P = vzorek $n–k+1$ bitů – předem stanovený dělitel
  chceme, aby T bylo dělitelné P, tj. aby T/P neprodukovalo zbytek



CRC pomocí aritmetiky mod 2, 2

- když $2^{n-k}D$ dělíme P dostáváme podíl Q a zbytek R a platí

$$\frac{2^{n-k}D}{P} = Q + \frac{R}{P}$$

  - ✓ zbytek je vždy o 1 bit kratší než dělitel, dělení je mod 2
  - ✓ zbytek se použije jako FCC, tj. $T = 2^{n-k}D + R$
- R zajišťuje , že T/P nemá zbytek

$$\frac{T}{P} = \frac{2^{n-k}D + R}{P} = \frac{2^{n-k}D}{P} + \frac{R}{P} = Q + \frac{R}{P} + \frac{R}{P} = Q + \frac{R+R}{P} = Q$$

  - ✓ sčítání je mod 2,
    součet dvou čísel mod 2 je 0,
    zbytek není, T je dělitelné P beze zbytku

Generující polynom G(x) = 10011 ($x^4 + x + 1$) a zpráva M(x) = 1101011011

Délka zabezpečení je rovna stupni generujícího polynomu, tj. k=4. Vypočteme zbytek po dělení $M(x) * x^4$.

```
11010110110000 : 10011 = 1100001010
10011
-----
010011
 10011
 -----
 0000010110
      10011
      -----
      0010100
        10011
        -----
          1110 = R(x)
```

Takže posíláme zprávu + zabezpečení $R(x)$, tj. $T(x)$= 1101011011 | 1110.

Nyní příjemce přijme zprávu i se zabezpečením, tj. $T(x)$ a vydělí $T(x)$ generujícím polynomem $G(x)$. Bude-li zbytek nula, zpráva byla doručena bezchybně.

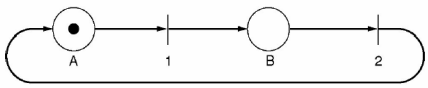
```
11010110111110 : 10011 = 1100001010
10011
-----
010011
 10011
 -----
 0000010111
      10011
      -----
      0010011
        10011
        -----
        000000 = R(x)
```

Takže žádná chyba nebyla detekována.

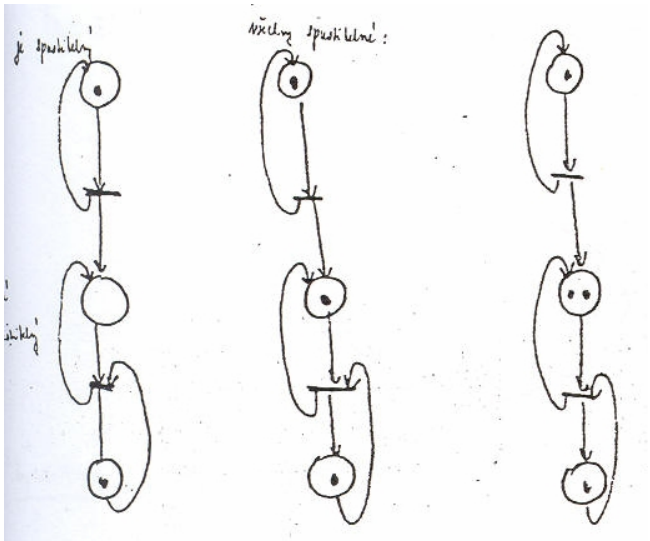
7. + 8. cvičení z UPS

Petriho sítě. Linkové protokoly, příklady.

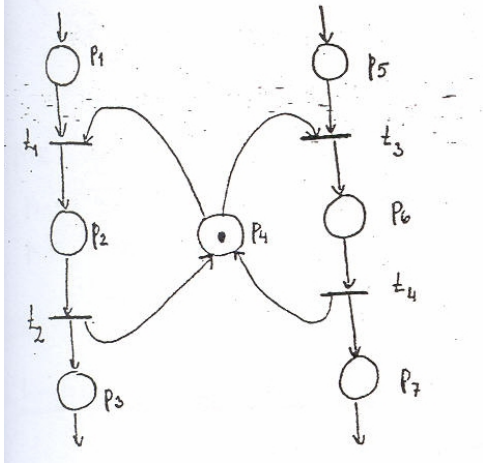
- Petriho sítě
  - Petriho síť je prostředek pro popis činnosti konečného automatu.
  - Bipolární orientovaný graf $N = (P, T, A)$
    - P ... neprázdná množina uzlů – míst (*places*)
    - T ... neprázdná množina přechodů (*transitions*)
    - A ... neprázdná množina orientovaných hran
  - Každá hrana spojuje uzly různých typů (P, T)



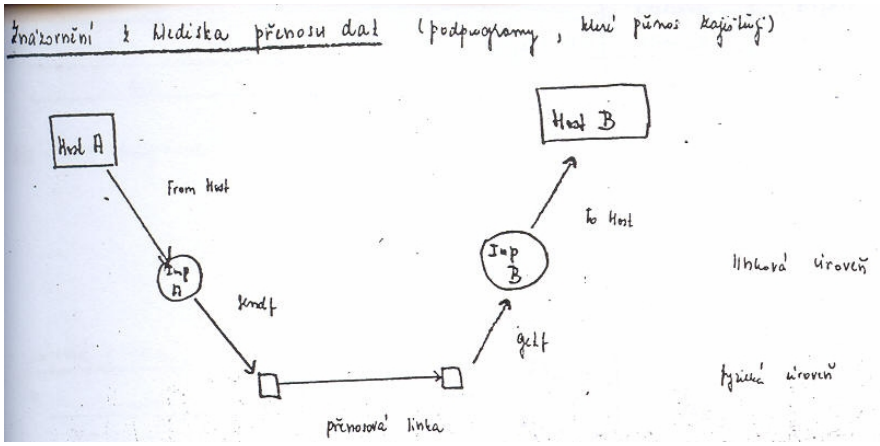
  - Značená Petriho síť je M = (P, T, A, μ), kde $\mu(p)$ je **ohodnocení**
  - Přechod je **dovolený**, pokud pro všechny jeho vstupy platí $\mu(p) > 0$
  - Dovolených přechodů může být více, vybíráme náhodně, který **spustíme**.
  - Po spuštění se ohodnocení předchozího uzlu snižuje o jedničku, ohodnocení následujícího uzlu zvyšuje o jedničku a zbytek je nezměněn.
  - Živá síť (z libovolného stavu lze přejít do stavu dalšího)
  - Zablokovaná síť (z nějakého stavu nelze přejít do dalšího)
  - Bezpečná síť je když: $\mu(p_i) \leq 1$, k-omezená pak: $\mu(p_i) \leq k$



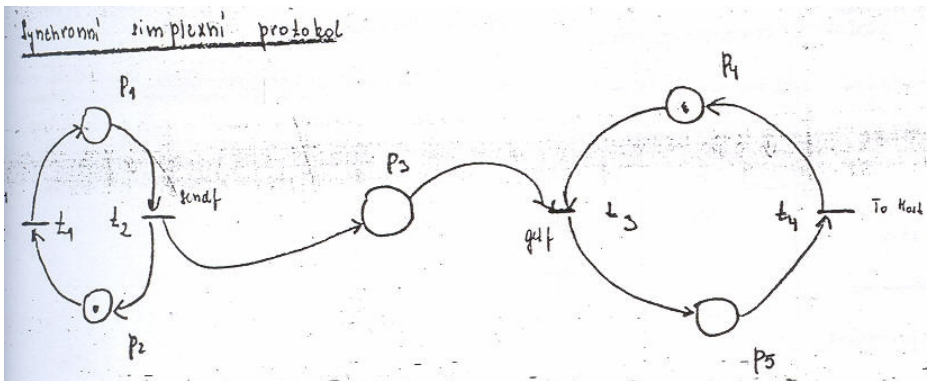
Řešení problému vzájemného vyloučení



Znázornění z hlediska přenosu dat (podprogramy, které přenos zajišťují)



Synchronní simplexní protokol



P_1, P_2 ... vysílač

P_3 ... kom. linka

P_4, P_5 ... přijímač

t_1 ... přenáší zprávy z nadř. vrstvy

t_2 ... odesílání zprávy do kom. linky

t_3 ... přenáší zprávy

t_4 ... zasílání zprávy do nadř. vrstvy

- Linkové protokoly

- Linková vrstva využívá služeb fyzické vrstvy a poskytuje své služby (přenos rámců) vrstvě síťové. Proto umí data přenášet pouze přes zařízení fyzické vrstvy (opakovače, HUBy, Tranceivery, apod.)

- Protokoly linkové úrovně:

    - Znakově orientované (BSC – *Binary Synchronous Control*)
        - přenáší data a řídicí znaky
        - tzv. charakter-stuffing (vkládání znaků)
        - řídicí znaky jako STX (*start*), ETX (*end*), DLE (*escape*)

    - Bitově orientované (HDLC – *High Data Link Control*)
        - přenáší data
        - tzv. bit-stuffing (vkládání bitů)
        - 01111110 na začátku i konci rámce
        - při odesílání je za každých 11111 je automaticky vložena 0, která je při příjmu automaticky odebrána
        - typy rámců: řídicí, informační (potvrzení), nečíslované (data)

- Protokoly pro řízení chyb:

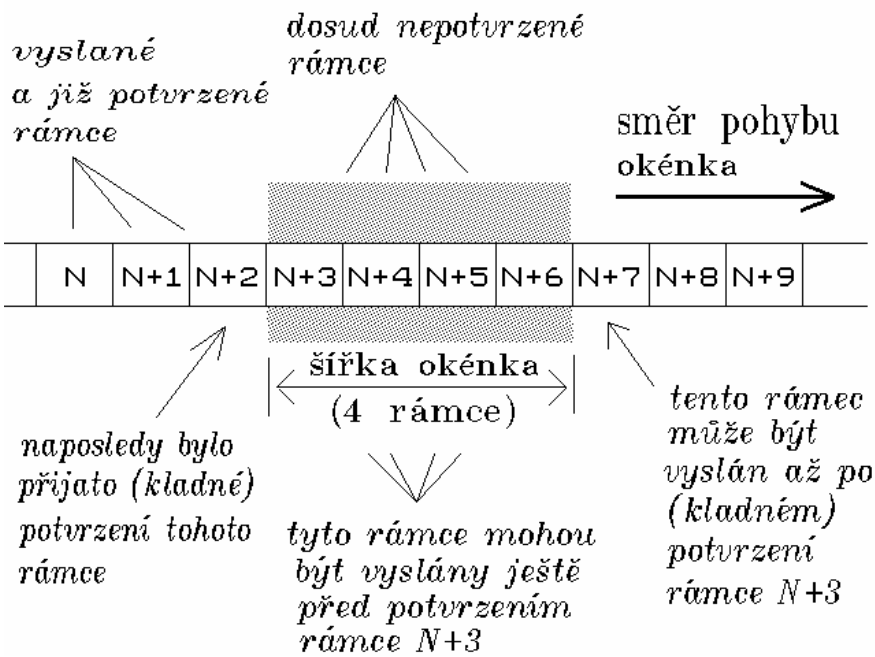
    - FEC (*Forward Error Control*) – redundance a samoopravné kódy
    - Stop&Wait
        - ARQ (*Auto Retransmission reQuest*) po *timeout*u
        - ACK (kladné potvrzování), NAK (záporné), obojí
        - Nevýhoda: na potvrzení se čeká po každém rámci
    - Selective Repeat
        - Znovu se vysílají jen NAK rámce
    - Go-Back-N
        - Znovu se vysílají všechny rámce od NAK (včetně již bezchybně přenesených)

    - Potvrzení
        - Individuální potvrzení – potvrzuje se každý rámec zvlášť
        - Skupinové potvrzení – potvrzuje celou skupinu přijatých nebo nepřijatých rámců

    - Potvrzování
        - Samostatné – zasílá se speciální rámec bez dat nesoucí pouze potvrzení
        - Piggybacking – potvrzení je součástí rámce nesoucího případná data

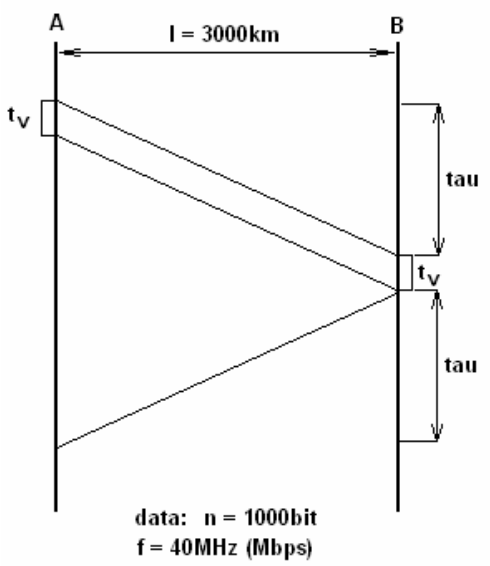
- Sekvenční a nesekvenční příjem dat
    - Nesekvenčnost může vzniknout chybami (najednou přijde znovu vyslaný chybně přijatý rámec)
    - Sekvenční příjem dat – data přijatá mimo sekvenci se zahazují a vyžaduje se opakování všech dat následujících – tj. Go-Back-N.
    - Nesekvenční příjem – využívá dostatečně velké okénko jako buffer, ve kterém si data přeorganizuje a zahodí až případné další duplicitní rámce – tj. vhodné pro Selective Repeat.

- Řízení toku dat (vysílač nesmí zahltit příjemce) – *Flow Control*
    - Řídí se především podle vstupních kapacit příjemce.
    - U protokolu Stop&Watt stačí příjemci nepotvrdit přijatý rámec.
    - U Selective Repeat nebo Go-Back-N (vysílají se rámce i když předchozí nebyly potvrzeny) stačí, aby vysílač vysílal pouze určitý maximální počet rámců.
        - Tím vzniká tzv. *okénko* vyslaných, ale nepotvrzených rámců.
        - Okénko se posouvá s každým přijatým potvrzením (proto tzv. posuvné okénko – *sliding window*).
        - Příjemce může okénkem posouvat jak se mu to hodí.



9. cvičení z UPS

- Zpoždění v síti a velikost okénka

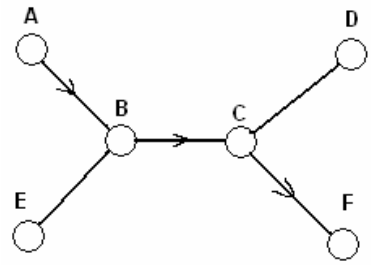


$t_v = n/f = 10^3/4*10^7 = 25$microsec.

$tau = l/v = 3*10^6/2*10^8 = 15$ milisec.

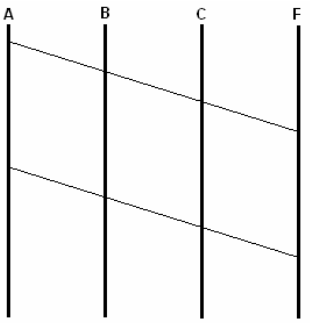
$t = t_v + 2*tau = 30.025$ milisec.

účinnost $= t_v/t = 25*10^{-6}/30*10^{-3} = 0.00166 =$
$= 0.2\%$

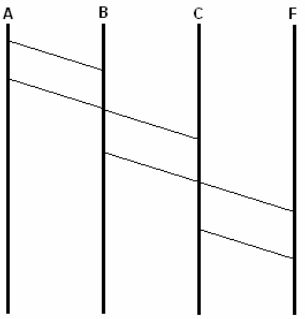


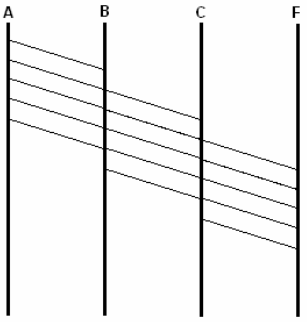
Režimy přenosu. Řízení přístupu.

- Přenos dat sítí
  - Naše topologie sítě.
  - Přepínání kanálů
    - Vytvoří se cesta (kanál, spojení), která se po dobu přenosu dat nemění.
    - Mezilehlé uzly všechny zprávy posílají od vysílajícího k příjemci s minimální režií.
    - Minimální zpoždění.



  - Přepínání zpráv
    - "Kanál" existuje pouze mezi sousedními uzly.
    - Každé dva sousední uzly si vyměňují celé zprávy.
    - Velké zpoždění.





- Přepínání paketů
  - Zpráva se rozdělí na několik paketů, které se postupně posílají sítí k příjemci.
  - Únosné zpoždění.

- Metody přepínání paketů

  - Store&Forward: Nejprve celý paket uloží a pak se rozhodne, co s ním (kudy poslat, smazat chybný, apod.)
  - Cut Through: Z hlavičky paketu rozhodne, kudy ho poslat a začne ho hned posílat (ještě než načte FEC, tj. než ví zda je rámec v pořádku).

♦ ISDN (*Integrated Services Data Network*)

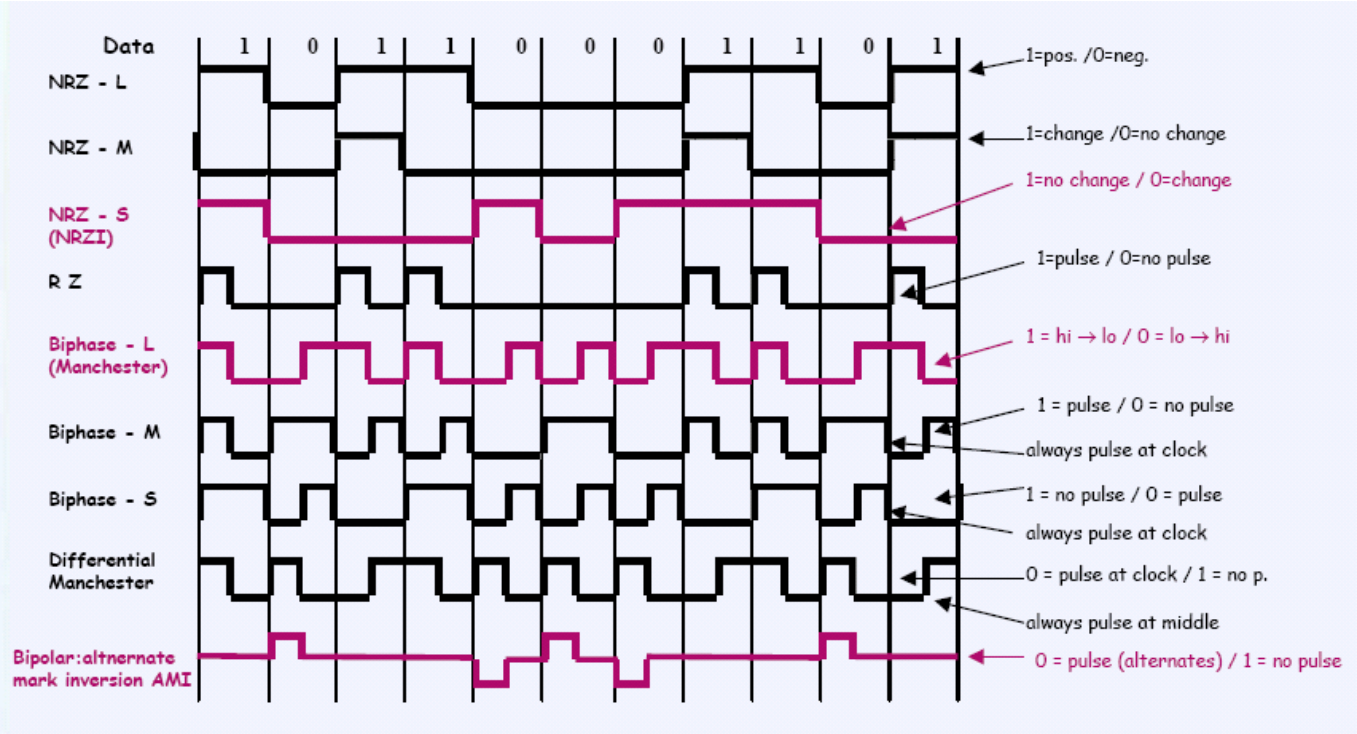
- Různé typy kanálů: A (analogový), B (digitální data 64kbps), C (řízení přenosu), D (řízení přenosu), E (vnitřní řízení ISDN), H (hybridní kanál)

- Varianty připojení:
  - Základní: 2B + 1D
  - Primární: 23B + 1D (USA, Japonsko)
    30B + 1D (Evropa)
  - Hybridní: 1A + 1C

- Multiplexování:
  - T1 (23B+1D, 1.544Mbps), T2 (4xT1), T3 (7xT2), T4 (6xT3, 274.176Mbps)
  - E1 (30B+1D, 2.048Mbps), E2 (4xE1), E3 (4xE2), E4 (4xE3), E5 (4xE4, 565.148Mbps)

♦ Přenos dat v režimu BASEband

- Jedná se o nemodulovaná data, obvykle nízké frekvence.



- Způsoby řízení přístupu ke sdílenému kanálu

  - Centralizované – jeden centrální uzel rozhoduje o přístupu k médiu
    - Výběr – stanice smí vysílat, pokud je nějakým algoritmem vybrána
    - Výzva – vyšle se výzva a pak se vyhodnotí došlé požadavky (v případě více požadavků rozhoduje priorita)

  Je zvolena jedna řídicí stanice (*master*), všechny ostatní stanice jsou řízené (*slave*). Ze všech řízených stanic se *výzvou* vybere jedna hlavní (*primary*) stanice – vysílač, která si k sobě zvolí jednu nebo více stanic vedlejších (*secondary*) – přijímače.

  - Decentralizované – neexistuje centrální uzel, který by řídil přístup

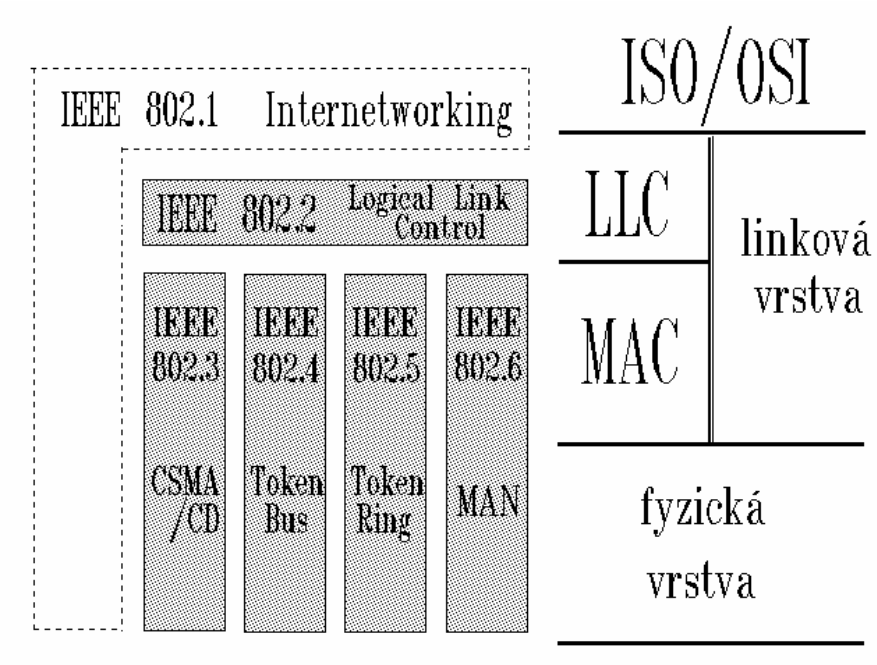
    - Náhodný přístup – stanice začne vysílat, kdy se jí to hodí. V případě konfliktu s jinou stanicí se nějakým algoritmem konflikt (zvaný kolize) vyřeší.

    - Rovnoměrný přístup – stanice si předáváním speciálního pověření dohodnou, které časové okénko bude která z nich vysílat.

    - Prioritní přístup – stanice vyšlou ostatním svoji prioritu, vysílat bude ta s největší prioritou.

- Rozdělení linkové vrstvy

  - Různé přístupové metody patří do dolní poloviny linkové vrstvy – MAC (*Media Access Control*). Jsou spíše blíže fyzické vrstvě.
  - Jsou zakryty vrstvou LLC (*Logical Link Control*), která plní ostatní ryze linkové záležitosti.

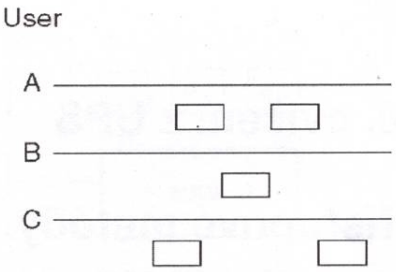


10. cvičení z UPS

Přístupové metody.

- Způsoby řízení přístupu ke sdílenému kanálu
  - Centralizované – jeden centrální uzel rozhoduje o přístupu k médiu
    - Výběr – stanice smí vysílat, pokud je nějakým algoritmem pomocí centrálního uzlu vybrána
    - Výzva – vyšle se výzva a pak se vyhodnotí došlé požadavky (v případě více požadavků rozhoduje priorita)
  - Decentralizované – neexistuje centrální uzel, který by řídil přístup
    - Náhodný přístup – stanice začne vysílat, kdy se jí to hodí. V případě konfliktu s jinou stanicí se nějakým algoritmem konflikt (zvaný *kolize*) vyřeší. Metody: prostá aloha, synchronní aloha, řízená aloha, CSMA, atd.
    - Rovnoměrný přístup – stanice si předáváním speciálního pověření dohodnou, kdy bude která z nich vysílat. Metody: protokol s binární mapou, MLMA.
    - Prioritní přístup – stanice vyšlou ostatním svoji prioritu, vysílat bude ta s největší prioritou.
- Centralizovaná metoda
  - Je zvolena jedna řídicí stanice (*master*), všechny ostatní stanice jsou řízené (*slave*). Ze všech řízených stanic se *výzvou* vybere jedna hlavní (*primary*) stanice – vysílač, která si k sobě zvolí jednu nebo více stanic vedlejších (*secondary*) – přijímače.
- Metody náhodného přístupu

  Využívají možnosti bezkonfliktního přenosu dat za podmínky, že možnost konfliktu není zcela vyloučena.
  - Prostá aloha
    - Stanice odešle rámec jakmile vznikne, nezajímá se o stav přenosového kanálu.
    - Příjemce zkontroluje zabezpečení a do doby 2τ přijatá data potvrdí. Jinak je rámec přeposlán znovu (po náhodné době).



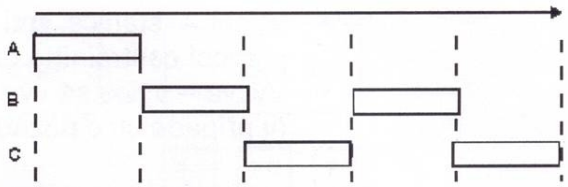
- Propustnost systému

  $S = be^{-2b}$

  Využitelnost kanálu 18%

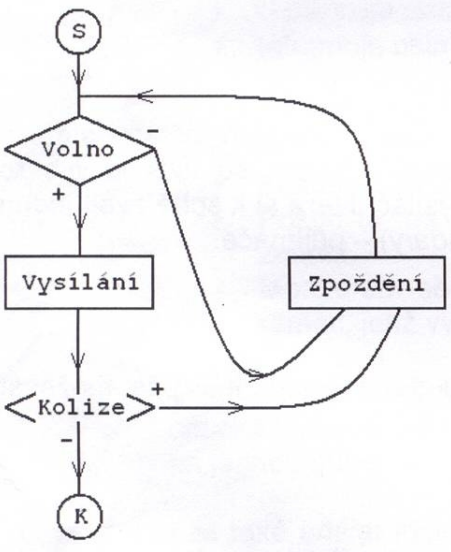
- Synchronní aloha
  - Stanice vysílají synchronně (na začátku časového slotu), čímž se vyloučí polovina konfliktů.
  - Propustnost systému

    $S = be^{-b}$



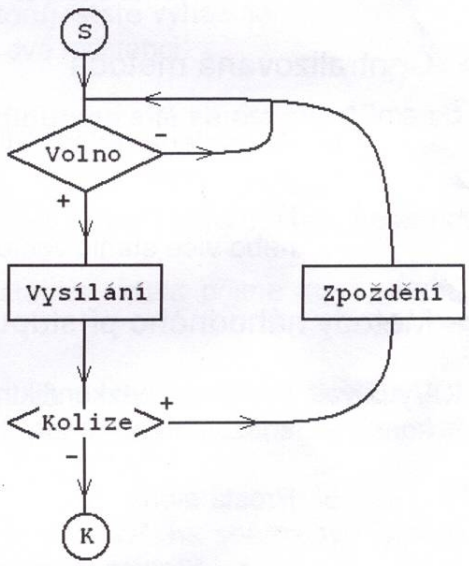
- CSMA (*Carrier Sence Multiple Access*)
  - Stanice naslouchá na kanále a přenos započne pouze, když je kanál volný.
  - Zabrání se většině kolizí, ale díky zpoždění v síti, ne všem.
  - Kolizi nerozpozná stanice, která vysílá hned. Ale až po timeoutu, když jí druhá stanice neodpoví.
  - Zpoždění opakování pokusu o přenos je náhodné.

1) CSMA – nenaléhající



Nemůže dojít k zahlcení systému.

2) CSMA – naléhající



Může dojít k zahlcení systému.

3) CSMA – *p*-naléhající: s pravděpodobností *p* bude naléhat a s (1 – *p*) nebude.

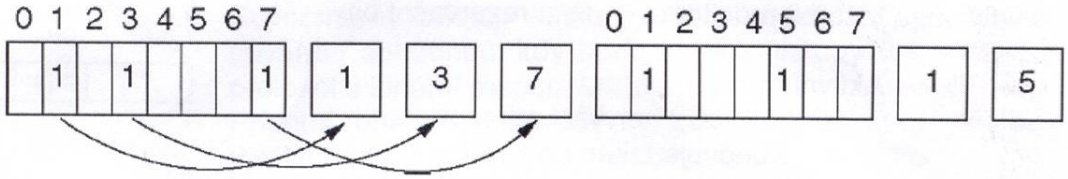
4) CSMA/CD (*CSMA with Collision Detection*):

- Jedná se o *p*-naléhající CSMA, ale...
- Stanice naslouchá i po započetí vysílání.
- Pokud data, která slyší odpovídají datům, která vysílá, je vše OK.
- Jinak se přeruší vysílání a zahájí se zotavení z kolize. To se provádí vysíláním tzv. signálu JAM po takovou dobu, aby byl slyšet všude na síti. Pak po náhodné době je pokus o přenos zopakován.
- Používá se v Ethernetu (IEEE 802.3).

♦ Metoda rovnoměrného přístupu

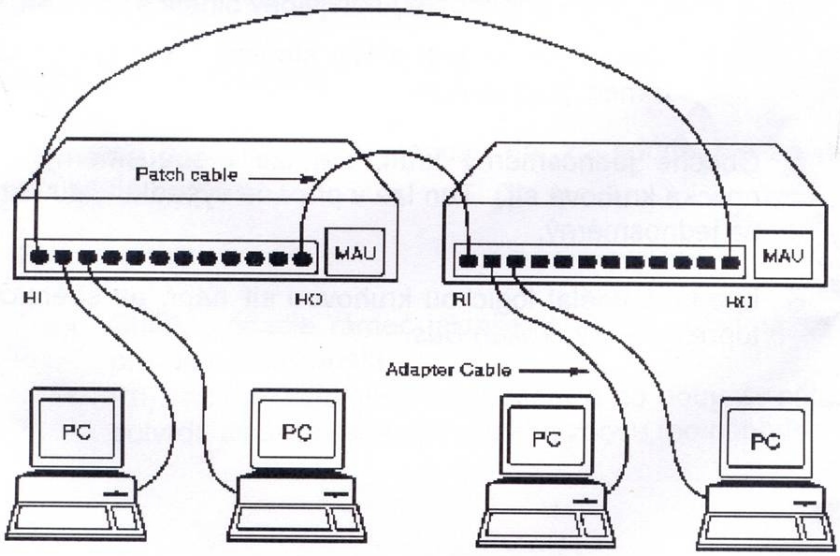
Vylučuje vznik kolize tím, že sestaví předem pořadí přístupu stanic ke sdílenému komunikačnímu kanálu.

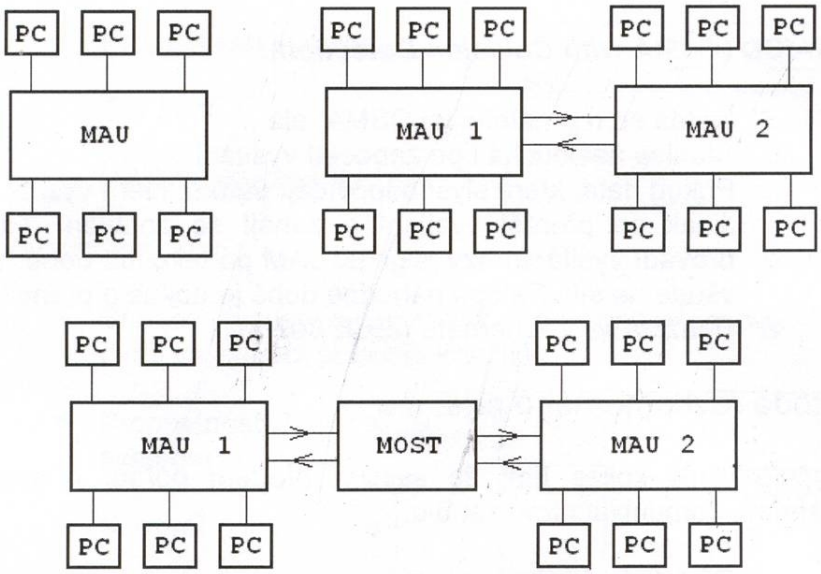
- Protokol s binární mapou
  - Řídicí stanice vyšle rámec s časovým okénkem, kde má každá stanice své místo a stanice se do něho zapíše a pak ví, kdy přesně bude vysílat.



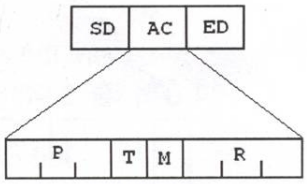
♦ Token Ring (IEEE 802.5)

- Kruhová síť s přístupovou metodou předávání pověření.
- Používá speciální HW – *Multistation Access Unit*.





- Oddělovací znaky: SD (*Start Delimiter*), ED (*End Delimiter*) jsou vytvořeny speciálními bity J, K (jiné než 1 nebo 0).



P ... priorita T ... příznak tokenu
M ... bit monitoru R ... rezervační bity

- Aktivní monitor:
  - Stanice s nejvyšší adresou.
  - Obnovuje token při ztrátě.
  - Kontroluje rámce (bit monitoru), staré vyhazuje.
  - Informuje ostatní stanice o své existenci.

- Stanice si vyměňují token, **rekonfigurace** sítě se provádí 10ms po ztrátě tokenu.
  - Každá stanice vyšle svoji prioritu stanici sousední.
  - Pokud ta má prioritu nižší, přepošle přijatý rámec, jinak ho zahodí.
  - Monitorem se pak stává stanice, která přijme konfigurační rámec, jenž vyslala.

- Obecně jednosměrný kruh, ale lze i obousměrný (např. FDDI optická kruhová síť). Ten lze v případě výpadku uzlu rekonfigurovat na jednosměrný.

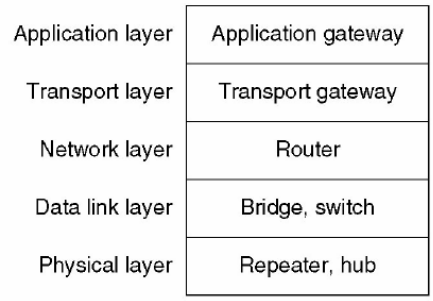
- Lze také udělat logickou kruhovou síť např. na sběrnicové fyzické topologii – tzv. *Token Bus*.

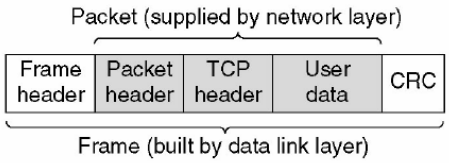
11. cvičení z UPS

Mosty, Spanning Tree algoritmus.

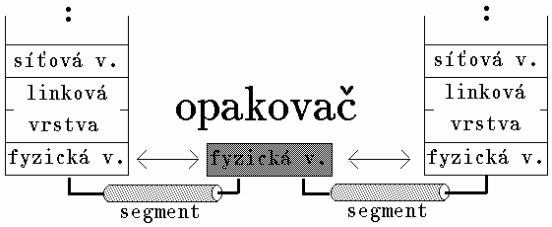
http://www.earchiv.cz/l209/slide.php3?&l=11&me=1

- Vzájemné propojování sítí
  - Počítačové sítě lze propojovat na různých úrovních ISO/OSI modelu (za pomoci různých zařízení)

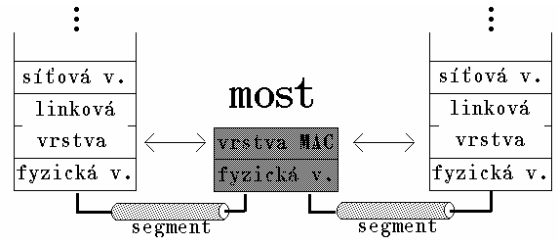




- Fyzická úroveň
  - Používaný HW: Opakovač (repeater) nebo rozbočovač (HUB) = víceportový opakovač
  - Zesílení signálu, převod signálu (koax. → UTP, fibre → koax.)
  - Propouští veškerý provoz (tj. veškerá data), nemá žádnou vyrovnávací paměť.
  - Může propojovat jen segmenty se stejnou přenosovou rychlostí.
  - Segmenty sítě propojené opakovači tvoří jednu tzv. **kolizní doménu** (*collision domain*), která končí u prvního mostu nebo směrovače. Všechny tyto segmenty patří také do jedné tzv.**všesměrové domény** (*broadcast domain*).



- Linková úroveň
  - Používaný HW: Most (*bridge*) nebo přepínač (*switch*) = = víceportový most
  - Pracují s adresami linkové úrovně (MAC či HW adresy), rozumí tedy struktuře rámce.



♦ Mosty a přepínače (obecně)

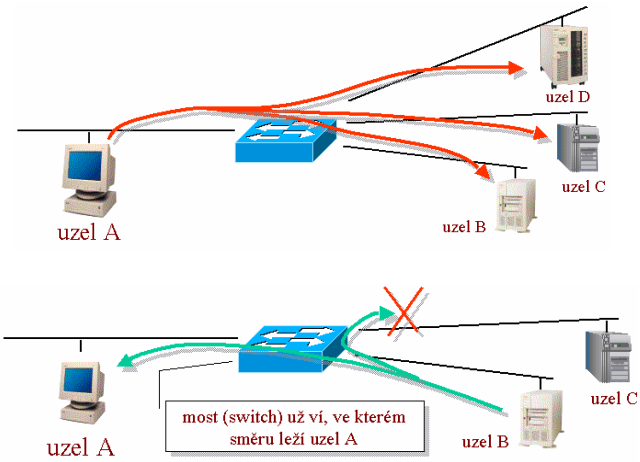
- Umí propojovat sítě s různou přenosovou rychlostí, protože obsahují vyrovnávací paměť.
- Zastavuje tzv. kolizní doménu (tj. nepropaguje kolize, signál JAM), ale segmenty sítě spojené přepínačem stále patří do jedné všesměrové domény, která končí až u směrovače.
- Přeposílání rámců buď *store&forward* nebo *cut through*.
- Speciální verze mostu, která zasahuje i na síťovou úroveň (tj. IP adresy) se označuje jako ***brouter*** (*bridge-router*).

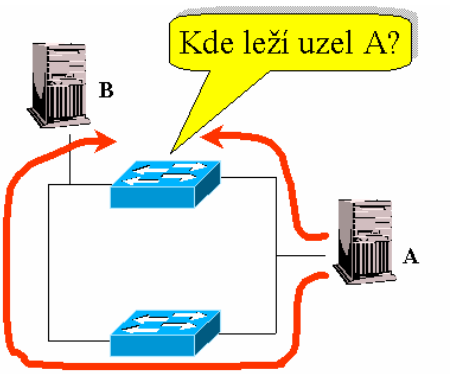
♦ *Ethernet*

- Funkce mostu: přeposílání, filtrování, učení a zapomínání.
- Mosty se buď „samoučí“ nebo mohou být staticky nakonfigurované.

- Most samouk se po zapojení do sítě chová jako opakovač (tj. neefektivně) až do doby, než se naučí informace o topologii sítě (tj. kde kdo je).

- Most (nebo přepínač) má pojmenované porty. Učí se tím, že z hlavičky příchozího rámce na určitý port si odvodí, kde se daná stanice nachází. Další dopravu pro danou stanici už nerozesílá do všech směrů, ale jen do jednoho naučeného.





- Samoučení může zkolabovat na smyčkách, pokud nejsou mosty chytré a neumí se domluvit.

- Pro domluvu používají STA (*Spanning Tree Algorithm*), což je algoritmus pro hledání tzv. kostry grafu.

- Po určité době mosty zapomínají naučené věci, čímž se dokáží vyrovnat s příp. změnou topologie sítě.

- *Token Ring*

    - V sítích *Token Ring* se používá tzv. *source routing*, kdy je rámec směrován k cíli informací předanou zdrojovým uzlem.
    - Informace je seznam uzlů, kterými musí rámec projít.

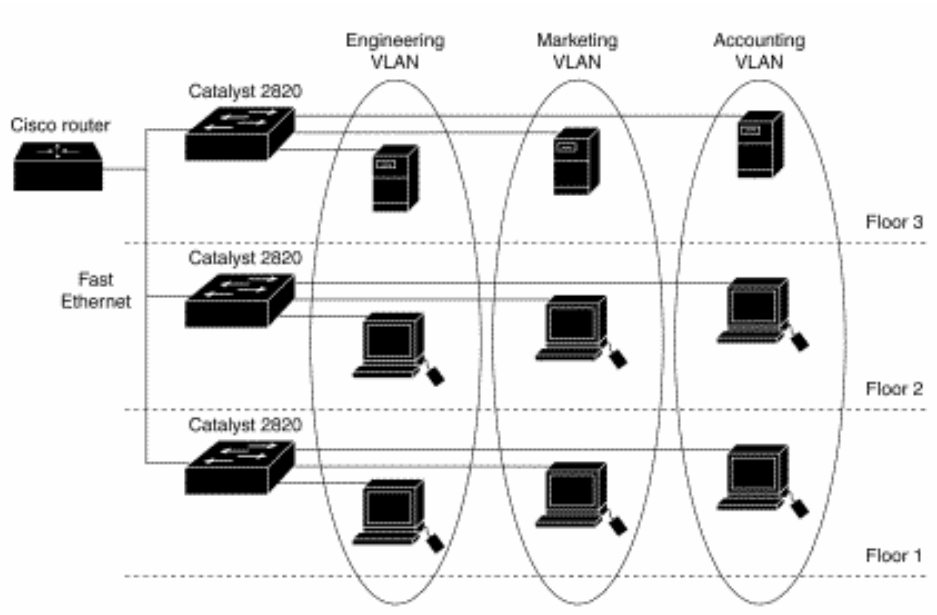
    - Tato informace je získána tzv. záplavovým směrováním.
        - Speciální rámec se šíří z každého uzlu do všech směrů kromě odkud přišel.
        - Dosáhne k cíli a pak se vrátí s potřebnou informací ke zdroji.

    - Jedná se sice o „směrování“, ale stále na linkové úrovni!

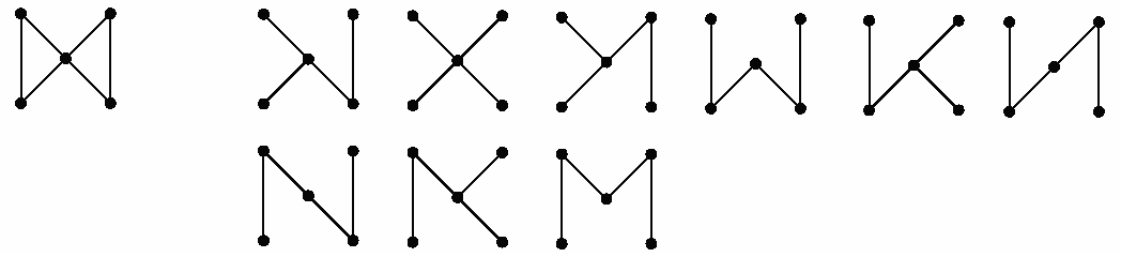
- VLAN (*Virtual Local Area Network*)

    - Různé počítače v různých segmentech sítě lze spojit do jedné virtuální LAN sítě.
    - Podporují to pouze lepší přepínače.
    - Přepínač přidá každému rámci do hlavičky tzv. *tag* označující VLAN, do které rámec patří. Rámce pak lze v přepínačích zasílat pouze do segmentů dané VLAN.

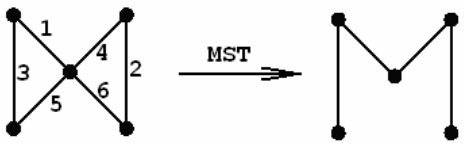
    - Stanice se seskupují na základě:
        - portu přepínače,
        - MAC adresy stanice,
        - komunikačního protokolu (např. IP, IPX lze ale i FTP, SSH),
        - masky podsítě.



- *Spanning Tree* algoritmus (hledání kostry grafu)
  - Kostra grafu má vždy N – 1 hran. Př: Graf a 9 jeho koster



  - Kostru grafu lze v síti vytvořit jednoduše pomocí tzv. záplavového adresování.
    - Každý uzel pošle zprávu do všech směrů, kromě toho, ze kterého ji obdržel.
  - ***Minimum Spanning Tree*** je pak minimální kostra grafu, tj. kostra grafu taková, že součet ohodnocení jejích hran je minimální.



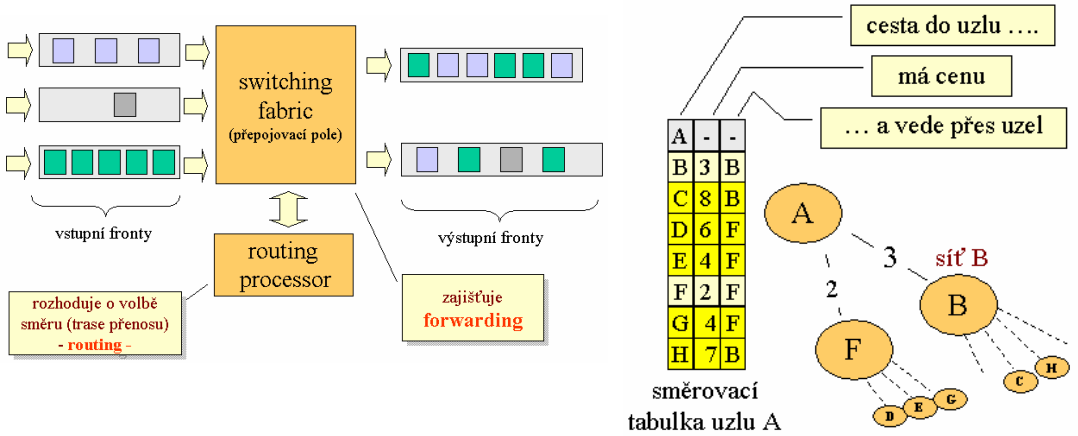
  - Jak se MST tvoří? Nejznámější algoritmus (nikoliv nejnovější nebo nejrychlejší) se jmenuje *Prim's algorithm*:
    - Náhodně zvolíme počáteční vrchol, ze kterého začneme tvořit MST.
    - Dokud existují vrcholy nepatřící do MST
      - Vybereme hranu mezi MST a zbytkem grafu s minimálním ohodnocením.
      - Vybranou hranu přidáme k MST.
  - Přepínače naslouchají i na neaktivních (nadbytečných) linkách. V případě chyby aktivní linky se ji pokusí nahradit nejvhodnější neaktivní.
  - http://students.ceid.upatras.gr/~papagel/project/prim.htm

12. cvičení z UPS

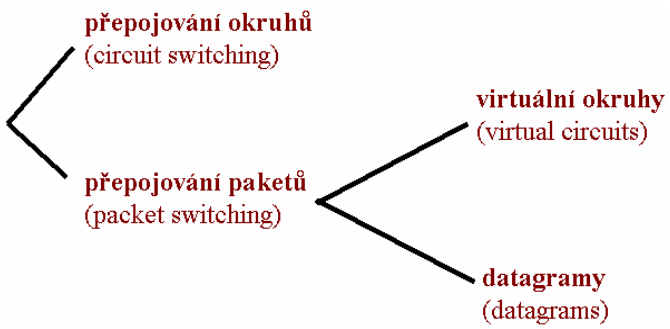
Směrování.

http://www.earchiv.cz/l209/nahled.php3?l=9&me=1&t=8x3

- Úkol síťové vrstvy
  - Poslední vrstva umístěná v síti. Zbytek ISO/OSI modelu je na koncových stanicích
  - Doručovat data od zdrojového PC až k cílovému, což může obsahovat skok přes mezilehlé uzly (směrovače).
  - Skok přes mezilehlé uzly:
    - Směrování (*routing*) – algoritmus pro určení cesty.
    - Přeposílání (*forwarding*) – vlastní přeposlání paketu.
- HW: Směrovač (*router*) – používá IP adresy, má 2 a více portů, pro směrování udržuje informace o topologii sítě v tzv. směrovací tabulce (*routing table*).



- Možnosti síťové vrstvy:



- Rozdíl mezi virtuálními okruhy a IP datagramy

    - Virtuální okruhy:
        - Zasílají pakety, které obsahují pouze ID virtuálního okruhu.
        - Všechny pakety putují stejnou cestou (mezilehlé uzly provádí pouze *forwarding*).
        - Je zaručeno správné pořadí jejich doručení.
        - Jsou **stavové**, tj. navázáním spojení přechází klient/server do jiného stavu.

    - IP datagramy:
        - Obsahují ID cílového uzlu.
        - Každý putuje jinou cestou (mezilehlé uzly provádějí jak *routing* tak *forwarding*).
        - Není zaručeno správné pořadí jejich doručení.
        - Jsou **bezestavové**, tj. klient/server odešle IP datagram, jeho stav se nemění.

- Algoritmy směrování

    - Neadaptivní
        - Nereaguje na změnu topologie sítě.
        - Uzly mezi sebou nepřenáší aktualizační informace.
        - Umí vypočítat optimální cestu předem.
    - Adaptivní
        - Reaguje na změnu topologie sítě.
        - Optimální trasy počítají průběžně.
        - Uzly si průběžně vyměňují aktualizační informace.

- Centralizované směrování

    - Založeno na existenci centrálního uzlu.
        - zná topologii sítě,
        - vypočítává optimální cesty,
        - informuje uzly o vypočtených hodnotách.

    - Kdykoliv zařízení směruje data a neví kudy, zeptá se centrálního uzlu a výsledek dotazu pak uchovává ve své vyrovnávací paměti.

- Izolované směrování

    - Směrování (rozhodování o volbě směru) provádí jednotlivé uzly samostatně.
    - Každý směrovač se rozhoduje jen podle sebe (proto „izolované“).

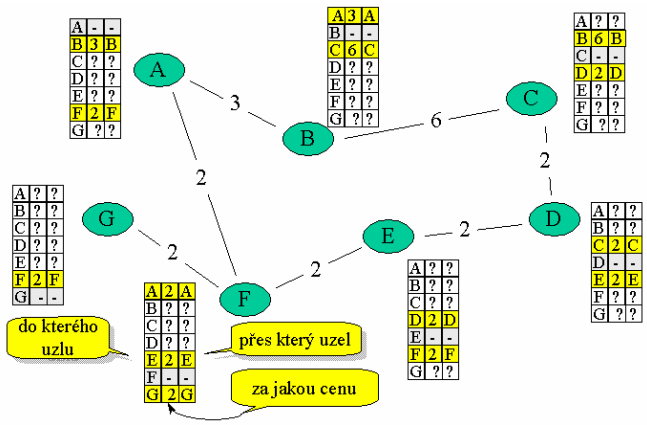
- Varianty
  - Záplavové směrování (*flooding*) – pakety se rozesílají v mezilehlých uzlech do všech směrů (kromě směru odkud přišly).
  - Metoda horké brambory (*hot potato routing*) – idea: když je zle (plné fronty), je potřeba se paketů zbavit co nejrychleji, tj. odeslat je skrze nejkratší frontu. Používá se pouze jako doplňková metoda (pro případ nouze).
  - Metoda zpětného učení (*backward learning*) – uzel se učí topologii z procházejících paketů. Pamatuje si cestu ke zdroji paketu (stejné jako mosty, jen s IP adresami) a její metriku.

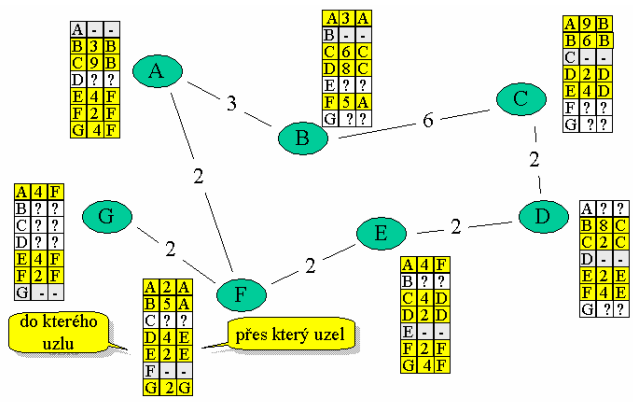
♦ Distribuované směrování

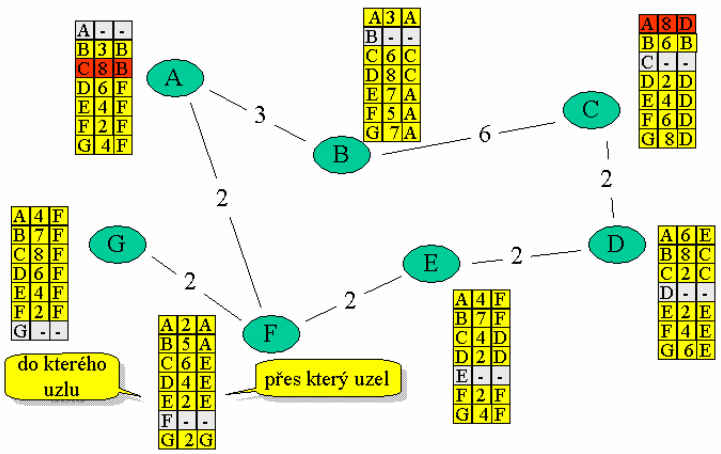
- Směrovače vzájemně spolupracují (vyměňují si aktualizační informace – směrovací tabulky).
- Dva základní přístupy (algoritmy)
  - DVA (*Distance Vector Algorithm*)
  - LSA (*Link State Algorithm*)

♦ DVA

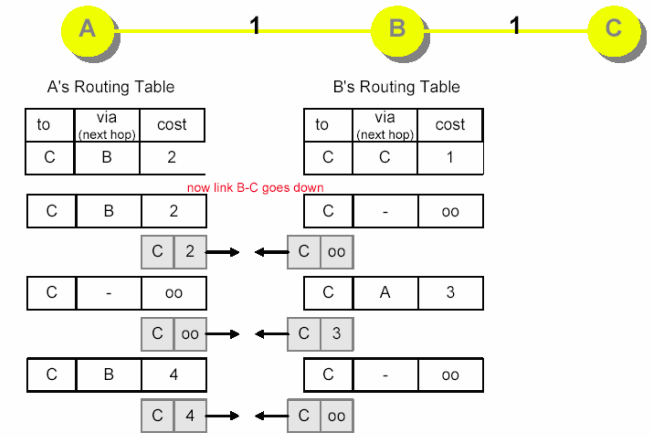
- Každý směrovač si udržuje tabulku min. vzdáleností od všech ostatních směrovačů.
- Směrovače si tyto tabulky vzájemně vyměňují.
- **Výměna probíhá jen mezi sousedními uzly.**
- Např.: RIP (Routing Information Protocol), IGRP (Interior Gateway Protocol)







- RIP používá tzv. *hop-count* metriku. Ohodnocení nekonečna je 16, takže se používá jen pro malé sítě. Směrovací tabulky rozesílá sousedům každých 30 sekund. Využívá služeb UDP/520.
  - Problém: Počítání do nekonečna (*Count to Infinity*)



  - Řešení: *Split Horizon* – směrovač nebude inzerovat zpátky, tj. jde-li o cestu do A, kterou se Y dozví od X, pak Y nebude cestu do A zpětně inzerovat uzlu X, od kterého ji získal.
  - Doplňek: *Poisoned Reverse* – uzel Y inzeruje zpět uzlu X dostupnost s hodnotou „nekonečno“.

♦ LSA

- Tyto algoritmy zasílají pouze tzv. *Trigerred Updates* (aktualizační informace pouze při změně ohodnocení nebo výpadku cesty).
- Např.: OSPF (*Open Shortest Path First*)

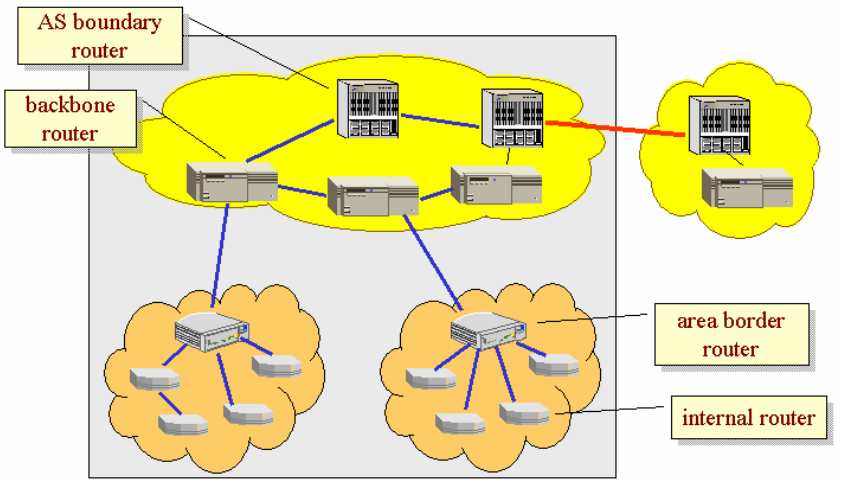
- Každý uzel má informaci o celé topologii a dělá veškerá rozhodnutí sám. Přibližná funkce:

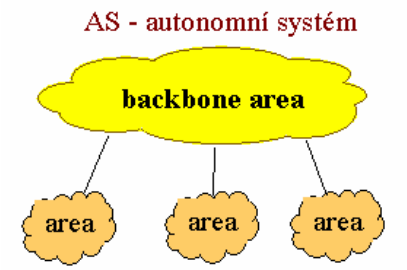
  1) Nejprve si každý uzel zjistí své přímé sousedy (pomocí paketů HELLO).
  2) Pak už průběžně zjišťuje pouze dobu jejich odezvy (pakety ECHO).
  3) Při změně uzel sestaví speciální paket a ten záplavovým směrováním odešle všem ostatním uzlům.

♦ Hierarchické směrování

- Obecně nejsou DVA ani LSA vhodné pro směrování ve velkých WAN, kterou bezesporu Internet je.

- Založena na myšlence rozložit velkou síť na více podsítí. Směrovací tabulky se nedostanou mimo podsíť. Každá podsíť má jen jeden I/O uzel (tzv. hraniční směrovač) pro komunikaci s ostatními podsíťěmi.





- Směrovače uvnitř oblasti znají dokonale danou oblast, všechno co tam nepatří směrují na hraniční směrovač (ABR).
- Hraniční směrovače (ABR) si mezi sebou pomocí páteřní oblasti vyměňují informace o dostupnosti svých podsítí.
- To, co nepatří do dané AS, je zasláno hraničnímu směrovači oblasti (AS BR), který data nasměruje do příslušné AS.

- Směrování uvnitř AS je *Intradomain Routing* a používá nějaký *Interior Gateway Protocol* (např. RIP, OSPF, IGRP nebo EIGRP). Naproti tomu směrování mezi AS je *Interdomain Routing* a využívá nějaký *Exterior Gateway Protocol* (např. EGP nebo BGP).

13. cvičení z UPS

Odevzdávání semestrálních prací.

- Odevzdávání semestrálních prací
    - E-mailem: Vlastní práce a odkaz na WWW dokumentaci.
    - Osobně: Papírová podoba dokumentace.
- Dotazy týkající se cvičení z UPS
- Dotazy týkající se vlastní tvorby semestrálních prací
- That’s all folks…