IP adresa, subnetting, supernetting, CIDR

Problém úbytku IP adres

Nedostatečná "granularita" (jemnost) při přidělování IP adres způsobuje poměrně velké plýtvání IP adresami. Síťové adresy třídy A, kterých je nejméně, se prakticky přestaly přidělovat, a u adres třídy B se s perspektivou možnosti jejich brzkého vyčerpání také rychle přešlo na úspornější systém přidělování. Když přišel někdo s žádostí o přidělení IP adres pro svou síť o 1000 uzlech, již nedostal 1 adresu třídy B, ale několik adres třídy C, například 8 (aby měl určitou rezervu pro růst své sítě, a nikoli jen 4, které by měly teoreticky stačit). Tím se místo 65 635 IP adres nenávratně vyčerpalo jen 8 x 256, tedy 2048 IP adres.

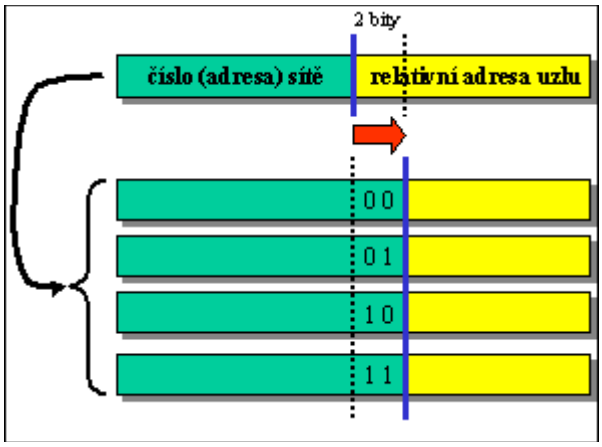
Problém nárůstu směrovacích tabulek

Přidělování několika síťových adres třídy C místo jedné adresy třídy B značně zpomalilo úbytek IP adres, ale na druhé straně akcelerovalo jiný nepříjemný problém - který by se projevil tak jako tak, ale nejspíše poněkud později. Jde o problém s objemem směrovacích informací, které musí být zpracovávány při rozhodování o volbě dalšího směru (neboli při faktickém směrování). Pro správné pochopení podstaty problému si představujme, že každý uzel, který provádí směrování (rozhoduje o volbě dalšího směru) neboli každý směrovač (router), má pro své rozhodování k dispozici tzv. směrovací tabulku (routing table). V té má pro každou síťovou adresu jednu položku, s údaji typu "pakety pro síť XY posílej takovým a takovým směrem". Přitom faktické směrování vyžaduje prohledávání takovéto tabulky, takže složitost směrování jako operace bude nutně záviset na velikosti směrovací tabulky.
Nyní si představme stejný příklad jako výše, se sítí, která má přibližně 1000 uzlů. Pokud by takováto síť dostala přidělenu jednu síťovou adresu třídy B (a tím "odčerpala" 65 636 konkrétních IP adres), pak by ve směrovacích tabulkách všech směrovačů stačila vždy jen jedna položka pro tuto síť. Pokud by při úspornějším přidělování IP adres zmíněná síť o 1000 uzlech dostala např. 8 síťových adres třídy C (čímž by "odčerpala" jen 2048 jednotlivých IP adres), pak by ve směrovacích tabulkách muselo být pro tuto síť vyhrazeno 8 položek, po jedné na každou síťovou IP adresu třídy C (a to obecně v každém směrovači v celém Internetu).
Podstata problému je zde v tom, že původní koncepce IP adres nedokáže nijak využít skutečnosti, že zmíněných 8 síťových adres třídy C "vede" na stejné místo, resp. nedokáže je spojit (tzv. agregovat) do jediné "společné" položky směrovací tabulky. Tím významně roste jak spotřeba místa v paměti, kde jsou směrovací tabulky uchovávány, tak i složitost práce se směrovací tabulkou a tím i vlastní směrování.
Prapůvodní příčina problému je přitom v tom, že nelze libovolně měnit pomyslnou dělící čáru, která vymezuje obě logické části 32-bitové adresy. Autoři protokolů TCP/IP pamatovali jen na tři "předdefinované" polohy této dělící čáry, odpovídající třídám A, B a C.

Masky IP adres

Cesta k možnému řešení se otevírá v okamžiku, kdy se k IP adresám připojí explicitní údaj o poloze pomyslné dělící čáry. Ačkoli by stačilo jednoduché číslo, udávající bitovou pozici příslušného předělu, v praxi se tato informace vyjadřuje nejčastěji ve formě tzv. masky - 32-bitového řetězce, který obsahuje 1 v těch bitech, které odpovídají síťové části adresy, a nulou tam kde jde o relativní adresu uzlu v rámci sítě. To sice dává prostor k možné "nespojitosti" obou logických částí, v praxi se však tato možnost nevyužívá.
Pomocí masky lze dosáhnout dvou různých efektů:
Lze spojit několik "sousedních" síťových adres do jedné síťové adresy - tomuto postupu se říká supernetting. Lze rozdělit jednu síťovou IP adresu na několik menších síťových adres - tomu se říká subnetting

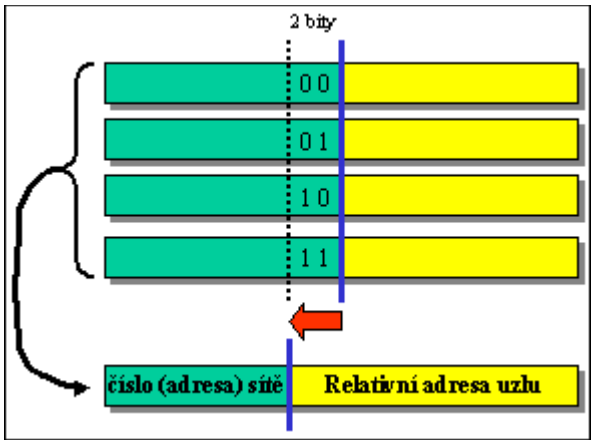
Subnetting a jeho použití



Technika tzv. subnettingu je starší a její obvyklé použití nevyžaduje žádnou změnu ve způsobu práce s IP adresami v globálním měřítku. Typicky se totiž tato technika používá v nějaké omezené oblasti, kde je třeba efektivněji využít již přidělených IP adres.
Uveďme si jako příklad firmu, která provozuje několik velmi malých lokálních sítí (například jednu pro své obchodníky, jednu pro účetní, jednu pro techniky atd.), přičemž počty uzlů v těchto sítích jsou malé nechť se dohromady vejdou do počtu 256 (resp. 254). Místo toho, aby zmíněná firma požadovala po jedné síťové adrese třídy C pro každou svou síť, díky subnettingu může vystačit s jedinou síťovou adresou třídy C pro všechny své sítě - pokud si ji pomocí vhodně nastavených masek rozdělí na tak velké části, jaké potřebuje.
Rozdělení jedné síťové adresy na několik síťových adres přitom odpovídá posunutí pomyslné dělící čáry mezi oběma složkami směrem k nižším bitům (doprava ve smyslu obrázků). Konkrétní posunutí je samozřejmě vždy definováno příslušnou maskou.
Důležitý je na subnettingu princip uzavřenosti - to, že rozdělení původní síťové adresy pomocí specifické masky resp. masek má pouze lokální platnost a nikoli platnost globální. Aplikováno na naši pomyslnou firmu s jedinou síťovou adresou třídy C to znamená, že způsob, jakým si ji "rozparceluje" pro své dílčí sítě, je její interní záležitostí, která je "viditelná" pouze uvnitř (v rámci dané firmy resp. jejích sítí) a není viditelná "zvenku" - mimo soustavu firemních sítí se nadále vše tváří jako jediná adresa třídy C. Lze se na to dívat také tak, že informace o detailním "rozparcelování" není šířena ven, mimo místa, kde vzniká a kde se třeba i mění. Z tohoto důvodu je nutné, aby sítě používající stejnou adresu rozdělenou pomocí subnettingu měly jediný společný vstupní bod - pokud by jich bylo více, pak by ve vnějším okolí chyběla potřebná informace o rozdělení adres, a bez této informace by nebylo možné volit mezi více možnými vstupy.

Supernetting

Supernetting je pravým opakem subnettingu v tom, že posouvá pomyslnou dělící čáru mezi oběma složkami IP adresy směrem k vyšším bitům (na obrázcích doleva). Tím vlastně spojuje (agreguje) několik původně samostatných síťových IP adres v jednu výslednou. Nemohou to ale být zcela libovolné síťové adresy, nýbrž jen takové, které se shodují v určitém počtu vyšších bitů své síťové části, a vyčerpávají všechny bitové kombinace v příslušném počtu nižších bitů své síťové části (viz obrázek). S jistou dávkou zjednodušení by šlo říci, že musí jít o "sousední" síťové adresy. Typickým příkladem situace, kdy může být použit supernetting, je výše citovaný příklad sítě s cca 1000 uzly, která místo jedné adresy třídy B dostane 8 (nejspíše potřebným způsobem "souvislých") adres třídy C. Posunem pomyslné dělící čáry o tři bitové pozice k vyšším řádům (na obrázcích doleva) - kvůli tomu, že 8 je 2 na 3 - lze technikou supernettingu z těchto 8 síťových adres udělat jedinou síťovou adresu.



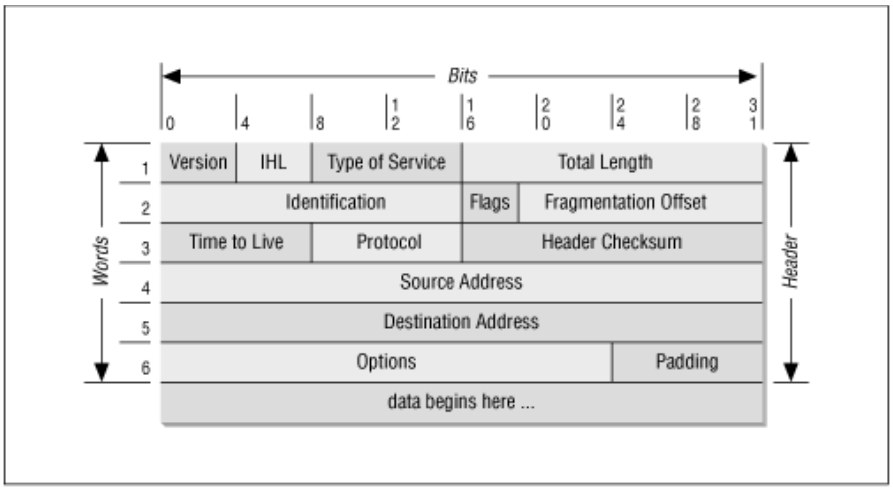
CIDR, alias: Classless InterDomain Routing
Techniku supernettingu lze využít k řešení výše naznačeného problému s nárůstem směrovacích tabulek v rozsahu celého Internetu. Aby to ale bylo možné, musí mít informace o příslušném "splynutí" (resp. o posunutí dělící čáry) globální charakter a být všude dostupná (na rozdíl od subnettingu, kde tato informace zůstává lokální).
Pro praktické nasazení supernettingu proto bylo nutné modifikovat celý koncept IP adres v rámci protokolů TCP/IP a způsob práce s nimi - místo tří předdefinovaných poloh "dělící čáry" se umožnila její libovolná poloha, s tím, že ke každé IP adrese musí vždy být k dispozici potřebná informace o konkrétní poloze této čáry. Jinými slovy ke každé IP adrese, ať již jednotlivé či síťové, musí být k dispozici její maska. V případě jednotlivé IP adresy (neboli adresy konkrétního uzlu) slouží tato maska k tomu, aby se z jednolitého 32bitového řetězce, který IP adresa představuje, mohla zjistit jeho síťová část a jeho relativní část identifikující příslušný uzel. U síťových IP adres pak maska udává jejich "velikost", měřenou v počtu jednotlivých IP adres přiřaditelných konkrétním uzlům sítě. Zde se však spíše hovoří o "bloku" a "velikosti bloku IP adres" (a místo masky v podobě bitového řetězce se používá jediné číslo, vyjadřující polohu dělící čáry jako počet bitů síťové části adresy).
Konkrétní "pravidla hry" o použití supernettingu, významu masek a IP adres i o manipulaci s nimi v rámci celého Internetu dostaly podobu konvence, pojmenované CIDR (Classless InterDomain Routing). Tato konvence vlastně nahrazuje původní "třídní" charakter IP adres (jejich rozdělení na třídy A, B a C - proto také přívlastek "classless"). Nyní se IP adresy přidělují po tzv. CIDR blocích, s velikostí danou příslušnou maskou - takže jemnost, s jakou jsou adresy čerpány z prostoru všech IP adres, může být velmi pružně přizpůsobována skutečným potřebám praxe, což dále snižuje tempo vyčerpávání celého adresového prostoru.
Důsledek CIDRu: adresy závislé na providerovi
V důsledku zavedení mechanismu CIDR se v Internetu významně změnila jedna důležitá věc. Dříve totiž nebyly IP adresy konkrétních sítí závislé na způsobu jejich připojení - pokud se provozovatel nějaké sítě rozhodl změnit svého providera, mohl si vybrat jiného a ponechat si své dosavadní IP adresy (neboť stačilo pouze změnit položky ve směrovacích tabulkách, odpovídající jeho síťovým adresám).
S nástupem mechanismu CIDR se však IP adresy staly závislé na způsobu připojení k Internetu resp. na konkrétním poskytovateli připojení (Internet providerovi). Ten totiž dostává přiděleny vždy celé tzv. CIDR bloky, ze kterých pak přiděluje IP adresy svým zákazníkům - ovšem detailní informaci o rozdělení CIDR bloku již nerozesílá do celého Internetu, ale ponechává ji jako lokální jen ve svých sítích. Mimo tyto sítě je v příslušných směrovacích tabulkách vždy jen jedna položka obsahující informaci ve smyslu: "k adresám, které spadají do CIDR-bloku XY vede cesta tudy a tudy". Díky tomu bylo možné značně zpomalit nárůst objemu směrovacích tabulek v celém Internetu - byť za cenu toho, že zákazník konkrétního providera při přechodu k jinému providerovi musí svou síť přečíslovat.

Protokol IP, format zahlavi, fragmentace, TTL, ToS, parametry

Věnujme se nyní způsobu, jakým je v TCP/IP modelu implementována výše zmíněná nespolehlivá a nespojovaná přenosová služba na úrovni síťové vrstvy. Definuje ji protokol, který se plným jménem nazývá **Internet Protocol**, ale nejčastěji je označován zkratkou IP.

IP protokol definuje základní jednotku dat, která je soustavou TCP/IP sítí přenášena na úrovni síťové vrstvy, tzv. **Internet datagram**, zkráceně **IP datagram**, a jeho přesný vnitřní formát. Definuje také způsob, jakým mají být jednotlivé IP datagramy směrovány, a toto směrování pak příslušný IP software také zajišťuje. Kromě toho IP protokol také podrobněji definuje i další aspekty poskytované nespolehlivé a nespojované přenosové služby - například podmínky, za jakých mohou být přenášené pakety zahazovány, kdy mají být generována chybová hlášení a jaká tato hlášení mají být.

IP datagram a jeho formát



Podobně jako každý jiný druh paketu či rámce, má i IP datagram dvě základní části: řídící část, tvořenou hlavičkou datagramu, a datovou část. Při vlastním přenosu se tento datagram vkládá (jako data) do datové části rámce, se kterým pracuje bezprostředně nižší vrstva - vrstva síťového rozhraní. V hlavičce jsou pak různé řídící informace, potřebné pro doručení datagramu resp. rámce - v případě hlavičky rámce jde mj. o fyzickou adresu skutečného odesilatele a bezprostředního příjemce, zatímco v hlavičce IP datagramu jde o IP adresy koncového příjemce a původního odesilatele. Jelikož v datovém rámci mohou být v principu přenášeny i jiné druhy paketů, musí zde být vyjádřeno také to, o jaký konkrétní druh paketu se v daném případě jedná (například v sítích typu Ethernet se druh paketu udává pomocí dvoubytové položky v hlavičce rámce, a IP datagramy zde mají vyhrazen kód 0800H). Podobně je tomu i na úrovni paketu, resp. IP datagramu - také v jeho hlavičce musí být vyjádřeno, jaký je význam obsahu datové části. V případě IP datagramu jde o jednobytovou položku, udávající číslo protokolu na úrovni transportní vrstvy, kterému obsah IP datagramu patří

Maximální délka IP datagramu
Za zmínku stojí například velikost položky TOTAL LENGTH, která udává celkovou délku IP datagramu (tj. jeho hlavičky i datové části), měřenou v oktetech (tj. v osmibitových bytech). Jelikož tato položka má rozsah 16 bitů, omezuje tím maximální možnou velikost IP datagramu na 65 535 oktetů.

Doba života
Další zajímavou položkou je jednobytová položka TIME TO LIVE. Ta udává, jak dlouho (měřeno v sekundách) se daný IP datagram může nacházet v soustavě vzájemně propojených sítí. Tato položka je jakousi pojistkou proti nekonzistentnostem ve směrovacích tabulkách, díky kterým by mohlo dojít k situaci, že by IP datagram byl směrován v kruhu, a obíhal v něm nekonečně dlouho. Tomu zabraňuje právě tato položka, jejíž obsah je každá mezilehlá brána povinna snižovat podle toho, jak dlouho se v ní příslušný datagram "zdrží". Jakmile dojde k vynulování položky TIME TO LIVE (doslova: doba života), je podle pravidel IP protokolu brána oprávněna daný IP datagram zahodit.

Fragmentace datagramů
Skutečnost, že IP protokol musí být schopen přenášet své datagramy prostřednictvím různých druhů přenosových cest, má některé významné důsledky. Například velikost datových rámců, přenášených na úrovni vrstvy síťového rozhraní TCP/IP modelu, je závislá na konkrétní přenosové technologii, pomocí které je daná dílčí síť realizována.
V případě lokálních sítí typu Ethernet je to 1500 oktetů, zatímco například sítě Token Bus připouští rámce až do velikosti 8191 oktetů, a veřejné datové sítě na bázi doporučení X.25 pracují s rámci až do velikosti 4096 oktetů. Některé moderní přenosové technologie však pracují s mnohem menšími rámci - například jen 128 oktetů či ještě méně. Protokol IP se ale dost dobře nemůže přizpůsobit nejmenšímu možnému formátu rámce, aby do něj mohl vždy vložit svůj IP datagram celý. Proto musí počítat s možností fragmentace, při které pro potřeby přenosu dochází k rozdělení původního datagramu na několik dílčích fragmentů - tak velkých, aby se již vešly celé do těch rámců, které je příslušná síť schopna skutečně přenést Jde přitom o tzv. netransparentní fragmentaci, při které jednotlivé fragmenty skládá do původního celku až jejich koncový příjemce. Ten pak k tomu využívá položky IDENTIFICATION, FLAGS a FRAGMENT OFFSET jednotlivých fragmentů.
(Aby byl cílový uzel schopen složit originální datagram, musí mít dostatečný buffer do něhož jsou jednotlivé fragmenty ukládány na příslušnou pozici danou offsetem. Složení je dokončeno v okamžiku, kdy je vyplněn celý datagram začínající fragmentem s nulovým offsetem a končící segmentem s příznakem "More Data Flag" nastaveným na False.)

Option-Type

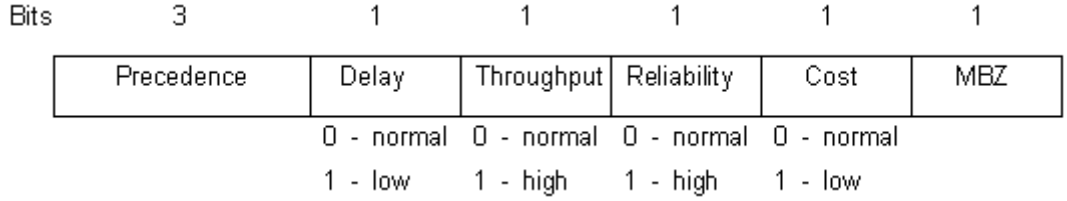
- **Copied Flag** - **0** indicates that the option is NOT to be copied to each fragment if the datagram is fragmented. A **1** indicates that the option IS to be copied.
- **Option Class** - **0** is used for **Control** (used normally) and **2** is used for debugging and measurement used for the **Internet Timestamp** option.
- **Option Number**
  - **0** - Special case indicating the end of the option list, in this case the option field is just one octet as no length or data fields are present.
  - **1** - **No Operation**, again the option field is just one octet with no length or data fields.
  - **2** - **Security** the length is 11 octets and the various security codes can be found in RFC 791.
  - **3** - **Loose Source Routing** which is IP routing based on information supplied by the source station where the routers can forward the datagram to any number of intermediate routers in order to get to the destination.

    - **4** - **Internet Timestamp**
    - **7** - **Record Route** records the route that a datagram takes.
    - **8** - **Stream ID** has a length of 4 octets.
    - **9** - **Strict Source Routing** which is IP routing based on information supplied by the source station where the routers can only forward the datagram to a directly connected router in order to get to the next hop indicated in the source route path.
- **Option-Length** - variable and not present for the NOP and the end of Option List
- **Option-Data** - variable and not present for the NOP and the end of Option List. See RFC 791 for the detail on the data content for each of the Options.

IP Options are not often used today, you may come across IP source-routing (loose or strict) on Unix machines and the like, perhaps for load balancing traffic where modern routing protocols are not being used.

Type of Service (TOS) Field

The following diagram illustrates the TOS field in detail:



Precedence - The following table details the precedence bits and their possible values:

- **000** (0) - Routine
- **001** (1) - Priority
- **010** (2) - Immediate
- **011** (3) - Flash
- **100** (4) - Flash Override
- **101** (5) - Critical
- **110** (6) - Internetwork Control
- **111** (7) - Network Control

Now the TOS bits themselves:

- **Delay** - when set to '1' the packet requests low delay.
- **Throughout** - when set to '1' the packet requests high throughput.
- **Reliability** - when set to '1' the packet requests high reliability.
- **Cost** - when set to '1' the packet has a low cost.
- **MBZ** - checking bit.

The thing to remember with the TOS bits is that bits set to 1 basically help speed up the packet flow.

Type of Service (TOS) has never really been used despite being part of TCP/IP for a long time. It has three parameters, **Delay**, **Throughput** and **Reliability**. **TOS Application Routing** is supported by OSPF and IS-IS but no application really supports it

DHCP

DHCP

Z Wikipedie, otevřené encyklopedie

Skočit na: Navigace, Hledání

DHCP (Dynamic Host Configuration Protocol) je aplikační protokol z rodiny TCP/IP. Používá se pro automatické přidělování IP adres koncovým stanicím v síti.

Současně s IP adresou posílá server stanicím (klientům) další nastavení potřebná pro používání sítě jako je adresa nejbližšího směřovače, masku sítě, adresy DNS serverů. Ve větších sítích se správce někdy rozhodne

posílat i adresy doporučených NTP, WINS, SMTP serverů, stárnutí ARP cache a jiné. Navíc je možné definovat i uživatelské parametry. Parametry, kterým klient nerozumí, ignoruje.

DHCP protokol přináší několik výhod:

- uživatelé si na počítači v souvislosti s připojením k síti nemusí nic nastavovat
- zaručuje, že se na síti nevyskytnou dvě stejné IP adresy (tzv. konflikt IP adres)
- správce sítě může „přečíslovat“ síť nebo změnit vlastnosti sítě s minimálním zásahem do práce uživatelů

DHCP protokol je rozšířením staršího BOOTP protokolu, který přiděloval IP adresy na neomezenou dobu. DHCP je s BOOTP obousměrně kompatibilní. To znamená, že DHCP klienti dovedou získat nastavení z BOOTP servru a DHCP server může přidělit IP adresu BOOTP klientovi (zde je třeba opatrnosti, protože BOOTP klient bude jednou přidělenou IP adresu používat už navždy).

[editovat]

Chování

Klienti žádají server o IP adresu, ten u každého klienta eviduje půjčenou IP adresu a čas, do kdy ji klient smí používat (*doba zapůjčení*, angl. *lease time*). Poté co vyprší, smí server adresu přidělovat jiným klientům.

Klient komunikuje na UDP portu 68, server naslouchá na UDP portu 67.

Po připojení do sítě klient vyšle broadcastem DHCPDISCOVER paket. Na ten odpoví DHCP server paketem DHCPOFFER s nabídkou IP adresy. Klient si z (teoreticky několika) nabídek vybere jednu IP adresu a o tu požádá paketem DHCPREQUEST. Server mu ji vzápětí potvrdí odpovědí DHCPACK.

Jakmile klient obdrží DHCPACK, může už IP adresu a zbylá nastavení používat.

Klient musí před uplynutím *doby zapůjčení* z DHCPACK obnovit svou IP adresu. Pokud lhůta uplyne aniž by dostal nové potvrzení, klient musí IP adresu přestat používat.

Protokol definuje roli i tzv. DHCP *relay agenta*. Používá se v situaci, kdy existují dvě nebo více sítí oddělené směrovačem a jen jedna síť má server. V takovém případě správce na směrovači zapne relay agenta a nastaví jej tak, aby všesměrové (*broadcast*) DHCP dotazy ze sítí bez DHCP serveru přeposílal do té sítě, která ho má. Agent k přeposílanému dotazu přidá masku té sítě, kde klienta zaslechl, aby DHCP server poznal, ze kterého adresního rozsahu má klientovi adresu přiřadit.

Parametry

Dialog Rozsah IP adres umožňuje zadání základních DHCP parametrů, které budou klientům přidělovány:

- Výchozí brána — musí být uvedena IP adresa směrovače, který je výchozí branou pro subsíť, z níž jsou IP adresy přidělovány (tzn. IP adresa rozhraní, ke kterému je daná subsíť připojena)! Výchozí brána v jiné subsíti nemá žádný smysl (byla by pro klienty nedosažitelná).
- DNS server — může být uveden libovolný DNS server (případně více DNS serverů oddělených středníky). Jako primární DNS server (tj. na prvním místě) však doporučujeme uvádět *DNS Forwarder* ve *WinRoute* (tj. IP adresu počítače s *WinRoute*). *DNS Forwarder* totiž dokáže spolupracovat s DHCP serverem (viz kapitola 5.3 DNS forwarder) a na dotazy na jména lokálních počítačů bude vždy odpovídat správnou IP adresou.

- WINS server

- Doména — lokální internetová doména. Pokud lokální doména neexistuje, pak tento parametr nenastavujte.

DNS

Co je to DNS

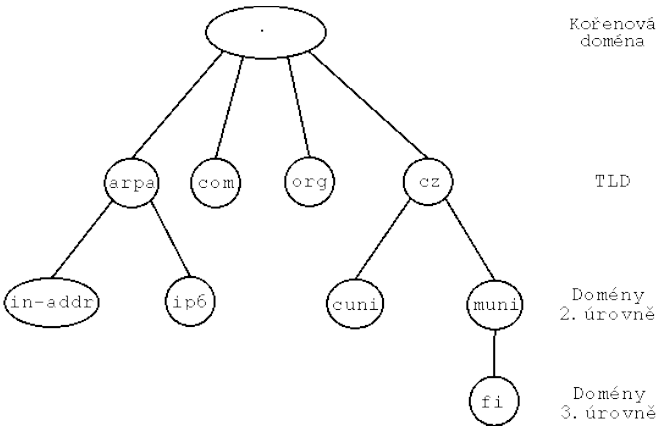
Současný Internet je veskrze postaven na bázi protokolové sady TCP/IP, která používá k jednoznačné identifikaci vzájemně komunikujících uzlů v síti IP adresu. Pro člověka jako uživatele je však snažší označovat počítače jménem a ne téměř nic neříkajícím 32 bitovým číslem nazývaným právě IP adresa (resp. 128 bitovým číslem, pokud uvažujeme protokol IPv6). Vznikl tedy systém DNS neboli Domain Name System, který nedokonalost lidského druhu elegantně obchází a který řeší i další, na první pohled ne zcela zřejmé, problémy související s překladem jmen na IP adresy a opačně.

Historie

První experimentální počítačová síť ARPANET vznikla koncem šedesátých let jako výzkumný projekt pod záštitou amerického úřadu DARPA (Defense Advanced Research Projects Agency) a propojovala důležité výzkumné organizace Spojených Států. V rámci tohoto projektu vznikla sada protokolů TCP/IP, která se záhy stala uznávaným standardem, a znamenala skutečný průlom v oblasti síťových technologií, neboť vedla ke vzniku dnešního Internetu.
A jak s tím souvisí systém DNS? Ano, již v době sítě ARPANET bylo zapotřebí překládat jména na IP adresy, jenže tehdy k tomu stačil jediný soubor. Jmenoval se HOSTS.TXT (dnes mu na Unixech odpovídá soubor /etc/hosts) a obsahoval tabulku hostitelů a příslušných IP adres. Obsah souboru spravovala organizace SRI's Network Information Center, která jej distribuovala po síti a to z jediného počítače! Jmenoval se SRI-NIC. A začaly vznikat problémy, na scénu totiž přišel TCP/IP a s ním i masívní rozšíření sítí za hranice malinkatého ARPANETu (tehdy několik stovek stanic). Zátěž na SRI-NIC tak začala být neúnosná a databázi hostitelů bylo stále obtížnější udržet v konzistentním stavu (duplicitní jména, rychlý růst sítě).
Pánové, jež se zasloužili o vznik ARPANETu, se však nevzdávali a pracovali dál. Z jejich společného úsilí vzešel nový a lepší systém, který řešil překlad jmen distribuovaně a ne centrálně jako v předešlém případě. Podporoval lokální správu dat, což rozložilo zátěž po celé síti a jména hostitelů byla uspořádána hierarchicky, což zamezilo vzniku duplicitních jmen. Vznikl tak systém DNS.

Jak DNS funguje

Systém DNS je celosvětově distribuovanou databází uchovávající záznamy o tom, která IP adresa patří ke kterému doménovému jménu, přitom IP adresa může mít doménových jmen i více. O databázi se starají programy zvané jmenné servery, které data z databaze poskytují klientům, tzv. resolverům. Jmenný prostor všech domén je uspořádán do hierarchické struktury podobné struktuře souborového systému na UNIXu. Databázi si lze představit jako strom s kořenovým uzlem, tzv. kořenová doména, která obsahuje všechny domény, na vrcholu, kde každý uzel má přiděleno nějaké jméno, tzv. doménu. Doménové jméno uzlu je seznam všech domén ležících na cestě mezi výchozím uzlem a uzlem kořenovým zapsaných zprava doleva v tečkové notaci. Doménové jméno tedy odráží příslušnost uzlu do určité domény, subdomény atd.



Doména

Doména je skupina jmen, které spolu logicky souvisejí např. tak, že mají společnou geografickou polohu, společnou přislušnost k nějaké organizaci nebo vytvářejí jistou síť. Domény lze dále členit na menší celky, tzv. subdomény.

Reverzní doména

V některých případech požadujeme po systému DNS tzv. zpětný překlad tzn. překlad IP adresy na reverzní doménu (doménové jméno). Pro tyto účely byla definována doména in-addr.arpa (v případě protokolu IPv6 ip6.arpa), která má subdomény 0 až 255. Domény jsou tvořeny IP adresami psanými v opačném pořadí. Např. síť 196.168.192.0/24 patří do domény 192.168.196.in-addr.arpa.

Doménové jméno

Doménové jméno vznikne spojením příslušných řetězců (domén) vzájemně oddělených tečkou, kde první řetězec je jméno počítače, druhý jméno domény, do níž počítač náleží atd. Takové jméno se pak čte zprava doleva. Celé jméno může být maximálně 255 znaků dlouhé, řetězec 63 znaků a může obsahovat pouze číslice, písmena a pomlčky (pomlčky nesmí být na začátku ani na konci řetězce). Např. uzel se jménem aisa.fi.muni.cz. je počítač se jménem aisa ležící v subdoméně fi subdomény muni domény cz kořenové domény (.). Poslední tečka na konci se uvádí z důvodu jednoznačného určení jména a představuje kořenovou doménu (může se téměř vždy vynechat). Takové doménové jméno se pak nazývá plně kvalifikované (FQDN, Fully Qualified Domain Name).
V kořenové doméně jsou definovány tzv. generické domény (TLD, Top Level Domains), např. org, com, edu, net, arpa, a dále dvouznakové domény jednotlivých států, např. cz pro Českou republiku.

Zóna

Správa domény může být z hlediska jejího rozsahu velice náročná. Jelikož systém DNS člení domény na menší celky, tzv. subdomény, je možné pověřit správou subdomény jiného správce. Hovoříme o tzv. delegaci domény. Zóna je pak část prostoru jmen domény, kterou obhospodařuje konkrétní správce. Je tedy tvořena doménou nebo její částí.

Resolver

Resolver je část systému, která dokáže nalézt IP adresu k příslušnému jménu a naopak. Je většinou implementován jako soubor knihovních funkcí, který se slinkuje s aplikací požadující tyto služby. Aplikace pak volá příslušné procedury, jako jsou např. gethostbyname(3) nebo gethostbyaddr(3), resolver zformuluje a pošle jmennému serveru dotaz a čeká na konečnou odpověď, kterou pak předá aplikaci.

Jmenný server

Jmenný server je právě zmiňovaný správce, který dohlíží na data definující příslušnou zónu a zajišťuje překlad jmen počítačů na IP adresy a naopak na žádost resolveru nebo jiného jmenného serveru. Podle uložení dat rozlišujeme následující typy jmenných serverů:

- **Autoritativní jmenný server** - každá zóna je pod správou alespoň jednoho jmenného serveru, který obsahuje kompletní data o zóně, tzv. autoritativní data. Tento server se označuje jako autoritativní. Je doporučováno, aby každá zóna měla nejméně dva servery tohoto typu.
- **Primární jmenný server** - je autoritativním jmenným serverem pro zónu a data o zóně získává z databází uložených na lokálním disku. Každá zóna má právě jeden primární jmenný server.
- **Sekundární jmenný server** - je autoritativním jmenným serverem pro zónu a data o zóně pravidelně kopíruje z databází primárního jmenného serveru (případně z jiných sekundárních jmenných serverů).
- **Caching-only jmenný server** - není autoritativním serverem (ani primárním ani sekundárním) pro žádnou zónu. Tento typ serveru pouze ukládá data do paměti, která jím procházejí.
- **Kořenový jmenný server** - je jmenný server obsluhující kořenovou doménu, přitom každý takový server je i primárním.
- **Stealth (tajný) jmenný server** - je autoritativní jmenný server, který je pouze uveden v konfiguraci jiného serveru, a není nikde zveřejňován. Může být primární i sekundární.
- **Forwarder (předávající) jmenný server**

Pozn.: Aby jmenný server správně fungoval, musí znát kořenové jmenné servery, které má uloženy v databázi na disku. Přitom ale není pro tyto data autoritou, považujeme se to za výjimku.

Dotazy (překlady)

Resolver je klientem systému DNS a zastupuje aplikaci požadující služby DNS. Resolver tak zformuluje dotaz, pošle jej místnímu jmennému serveru a očekává jeho odpověď. Zná-li náš server na příslušný dotaz odpověď (odpověď hledá ve svých autoritativních datech nebo v neautoritativních datech, které si uložil do vyrovnávací paměti z předchozích dotazů), zašle ji nazpět. Pokud ji nezná, kontaktuje další servery, přitom vždy začíná kořenovým jmenným serverem. Uvažujme např. dotaz na jméno aisa.fi.muni.cz. Kořenový jmenný server zjistí, že správou domény cz pověřil jiný server, pošle tedy našemu serveru IP adresu tohoto serveru. Náš server se obrátí na server domény cz a ten pošle adresu serveru domény muni.cz. Proces iterace pokračuje tak dlouho, dokud náš server nedostane IP adresu stroje aisa.fi.muni.cz (za předpokladu, že příslušné jmenné servery jsou funkční). Získané informace náš server ukládá do vyrovnávací paměti pro případ, že by se dotazy mohly opakovat.

- **Forwarder (předávající) jmenný server** - je jmenný server, který provádí rekurzivní dotazy za náš místní server. Místní server předá dotaz forwarder serveru a očekává konečnou odpověď, tváří se jako resolver.

Pozn.: Důležité jmenné servery, jako jsou servery generických domén nebo kořenové jmenné servery, nesmí podporovat rekurzivní dotazy z důvodu vyšší zátěže.

Protokol DNS

Protokol DNS je aplikační protokol využívající k transportu dat protokol UDP a TCP. K jednodušším dotazům, jako je překlad adres, se používá UDP. Pro odpovědi se používá UDP, ale pouze pokud je odpověď kratší než 512B. V opačném případě se použije pro přenos TCP. Protokol TCP se také používá pro přenos zón mezi primárním a sekundárním jmenným serverem. Jmenný server naslouchá dotazům na portu 53 (UDP i TCP).

Zdrojové záznamy

Autoritativní data zóny jsou uloženy v databázi ve formě tzv. zdrojových záznamů (RR, Resource Records). Každý záznam má přidělen typ, který popisuje druh dat v záznamu, a dále třídu, která vyjadřuje adresovací schéma sítě např. IP adresám odpovídá třída IN, adresám sítí Hesiod třída HS. Záznamy RR se přenáší sítí protokolem DNS a používají se v konfiguračních souborech systému DNS.

Typ	**Význam**
SOA (Start Of Authority)	Každá zóna obsahuje právě jeden záznam SOA, viz Záznam SOA
A (A host address)	32 bitová IP adresa, viz Záznam A
NS (Authoritative name server)	Jméno autoritativního serveru pro zónu, viz Záznam NS
CNAME (Canonical name for an alias)	Alias pro doménové jméno, viz Záznam CNAME
PTR (Domain name pointer)	Doménové jméno pro reverzní překlad, viz Záznam PTR
MX (Mail exchange)	Priorita a doménové jméno poštovního serveru, viz Záznam MX
HINFO (Host information)	Popis hardwaru a softwaru, viz Záznam HINFO a TXT
TXT (Text string)	Textový popis, viz Záznam HINFO a TXT

SLOW START Algoritmus

Pomalý štart (slow start)

TCP protokol nezačne po nadviazaní spojenia posielať pakety plnou rýchlosťou výstupného rozhrania až po naplnenie veľkosti okna na strane príjemcu. Miesto toho posiela pakety pomaly tak, že pošle jeden paket, počká až kým mu príde potvrdzovacia správa ACK, na korú zareaguje vyslaním dvoch paketov do siete atď. Na každú prišlú potvrdzovaciu správu ACK pošle do siete dva ďalšie pakety. Rýchlosť vysielania sa postupne zvyšuje až po hodnotu, kedy sa začnú pakety strácať. Táto fáza sa nazýva fáza *pomalého štartu* (pomalý preto, lebo vysielanie paketov sa začína pomaly). Zvyšovanie rýchlosti vo fáze pomalého štartu sa deje exponenciálne.

Predchádzanie zahlteniu (congestion avoidance)

Pomalý štart je implementovaný v spolupráci s algoritmom *predchádzanie zahlteniu*, aj keď ide o dva nezávislé algoritmy. Tento algoritmus sa snaží predchádzať zahlteniu siete tým spôsobom, že tempo zvyšovania rýchlosti zmení z agresívneho exponenciálneho na pomalé lineárne. Algoritmus reaguje na stratu paketov znížením aktuálnej rýchlosti na polovicu a zmenou tempa zvyšovania rýchlosti z exponenciálneho na lineárne, aby sa pomaly blížil k bodu zahltenia.

Implemetácia

Pre každé spojenie sa pre stranu posielateľa dát popri veľkosti vysielacieho okna *snd.wnd* (čomu na strane prijímateľa odpovedá prijímacie okno) udržuje aj veľkosť tzv okna zahltenia (congestion window) *cwnd*. Zavádza sa pojem efektívne okno *wind,* ktoré je definované ako

$$wind = min\ (snd.wnd,\ cwnd)$$

Ďalej sa zavádza premenná *sstreshd* (slow start threshold – medzná hranica pomalého štartu), ktorá odďeľuje od seba fázu pomalého štartu a fázu predchádzania zahlteniu.

Význam premennej *cwnd* je udržiavať veľkosť okna, ktoré by sa malo zväčšovať postupne a tým by mala rásť aj prenosová rýchlosť spojenia. Má však rásť len do rozmerov, pri ktorých sa zistí zahltenie. Zahltenie sa zisťuje stratou paketov. Vtedy veľkosť okna *cwnd* klesne na jedna a premenná *sstreshd* sa nastaví na polovicu veľkosti okna, pri ktorom došlo k zahlteniu.
Význam efektívneho okna je v tom že obmedzuje rýchlosť na úroveň ktorá by mala vyhovovať aj obmedzeniam v sieti aj obmedzeniam na strane príjemcu.

Na dvojicu okien *snd.wnd*, *cwnd* sa dá pozerať aj nasledovne: *cwnd* je kontrola toku na strane posielateľa – posielateľ (sender) nikdy nevypustí do siete viac nepotvrdených dát ako si myslí, že sieť stíha a *snd.wnd* je kontrola toku na strane prijímateľa – nikdy neviem naraz spracovať viac dát ako je *snd.wnd* (na strane prijímateľa je to *rcv.wnd*)

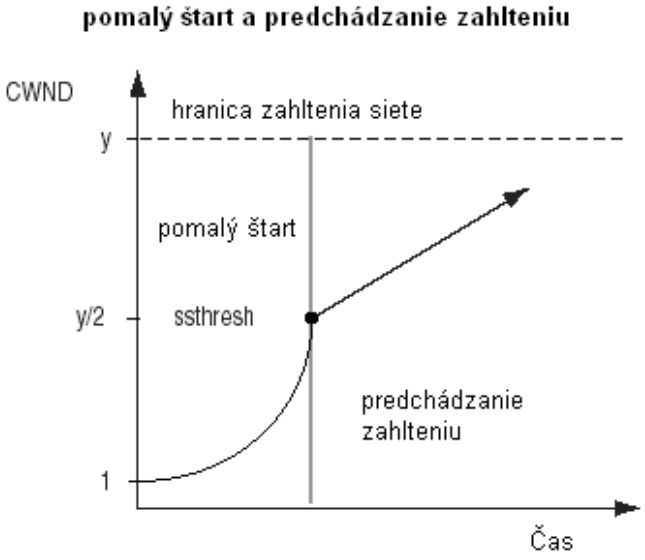
Formálnejšie je algoritmus navrhnutý takto[1]:

1) pri inicializácii spojenia sa nastaví *cwnd = 1* (segment), *sshtresh = 65535* (bajtov), resp. odpovedajúci počet segmentov
2) vysielateľ sa zaväzuje, že do siete nikdy nepošle naraz viac nepotvrdených dát ako je veľkosť *wind = min (snd.wnd, cwnd)*
3) ak detekujem zahltenie oboma metódami (t.j. vypršanie timera alebo duplikátny príchod takej istej potvrdzovacej správy ACK) tak nastavím *sshtresh = wind / 2* (segmenty, resp odpovedajúci počet bytov) (minimálne však 2 segmenty) a *cwnd = 1*
4) spojenie sa z hľadiska týchto dvoch algoritmov môže nachádzať v dvoch stavoch:
   - v stave pomalého štartu – to je za podmienky ak *cwnd* < *sshtresh*
   - v stave congestion avoidance – to je za opačnej podmienky *cwnd*>= *sshtresh*

5) podľa toho v akom stave je spojenie, tak sa zvyšuje hodnota okna cwnd nasledovne
   - pre pomalý štart cwnd = cwnd + 1 pre každú prijatú správu ACK
   - pre predchádzanie zahlteniu cwnd = cwnd + 1/cwnd pre každú prijatú správu ACK

Poznámky k algoritmu

- zaujímavý je vzťah medzi prijatím potvrdzovacej správy ACK a časom RTT (round trip time). Ak je veľkosť okna *cwnd = A*, tak v čase RTT by som mal prijať zhruba *A* potvrdzovacích správ ACK. To značí, že za čas RTT sa vo fáze prechádzanie zahlteniu zvýši veľkosť okna o 1 segment, kdežto pri metóde pomalého štartu o *A* (t.j. o toľko, koľko segmentov som poslal do siete).
- hodnoty *cwnd, sshtresh* sú v reálnych implemetáciách uvádzané v bajtoch, v schématickom popise sa pre prehľadnosť zvyknú uvádzať v segmentoch
- jedinou hodnotou ktorá sa zvyšuje postupne je hodnota *cwnd*. Hodnota *sshtresh* sa zvyšuje absolútne a to pri detekcii zahltenia. Jej úlohou je oddeliť od seba stav kedy sa zrýchľuje exponenciálne a kedy lineárne
- pomalý štart a predchádzanie zahlteniu nútia TCP protokol redukovať hodnotu *cwnd* na jedna, vždy keď sa detekuje strata paketu. Ak zahltenie siete trvá dlhší čas, množstvo premávky poslatej do siete klesá exponenciálne. Toto prispieva k vyprázdneniu front na smerovačoch a k odstráneniu zahltenia



RTP

RTP je protokol framework pro prenos dat. Kazdy RTP paket obsahuje hlavne payload type (identifikator co veze), sequence number (cislo inkrementovane s kazdym paketem, nahodne na zacatku session), timestamp (casove razitko payloadu, nahodne na zacatku session), SSRC (synchronization source ID), CSRC list (contributing source ID list). Krom RTP paketu se posilaji jeste RTCP pakety, RTCP paket je bud sender report (statistiky vysilacu), receiver report (statistiky prijimacu), source description (identifikace a popis session), nebo ridici paket. Vsechny maji hlavicku podobnou RTP paketum, reporty pak obsahuji casove razitko (v RTP i NTP formatu, aby se dal merit roundtrip a jitter), objem prenesenych dat, objem a procento ztracenych dat, jitter. No a to je v podstate vsechno, pak se uz jen rekne, ze vysilaci a prijimaci mohou RTP framework pouzit ke komunikaci, ze mohou definovat mixery (uzly, ktere spojuji vic vstupnich streamu pod vlastnim SSRC) a translatory (uzly, ktere modifikuji vstupni stream, ale nechavaji SSRC). Reakce na RTCP se nechava na aplikacich, pravdepodobne bude v podobe flow control a upravy sitovych parametru pomoci RSVP.

RSVP:

*** RSVP se výhradně signalizační protokol, který pro své šíření využívá informací získaných běžnými směrovacími protokoly.**

*** Zajištění QoS je založeno na explicitních rezervacích zdrojů ve všech routerech podél datového toku. Každý router tedy musí v paměti uchovávat potřebné stavové informace. Jedná se však o tzv. "měkký stav" (soft-state), neboť všechny tyto informace mají omezenou životnost a musí být periodicky obnovovány zprávami typu PATH a RESV. Informace a rezervace příslušející danému toku lze též zrušit explicitně pomocí zpráv typu PATHTEAR a RESVTEAR.**

*** Rezervace jsou iniciovány příjemcem a realizují se postupně proti směru datového toku. To je velmi výhodné zejména pro multicastové toky, protože se tím rozděluje zátěž spojená s instalací cest pro datový tok a umožňuje se též efektivní agregace rezervačních požadavků. Rezervace mohou též být heterogenní, tj. každý příjemce si stanoví vlastní parametry QoS.**

RTCP

Real-time Transport Control Protocol: RTCP

Jak bylo řešeno výše, RTP neposkytuje žádný mechanizmus na zajištění doručení, včasného doručení paketů, ani pro doručení paketů ve správném pořadí. Doručování paketů je monitorováno pomocí podpůrného řídicího protokolu RTCP. Tyto dva protokoly jsou často brány dohromady a označovány jako RTP/RTCP.

Řídicí protokol pro přenos v reálném čase (*RTCP, Real-time Transport Control Protocol*) spolupracuje s protokolem RTP. Používá periodické vysílání paketů od každého účastníka relace RTP všem ostatním účastníkům za účelem řízení výkonnosti a pro diagnostické účely. RTCP pomáhá RTP monitorovat doručení dat v rozsáhlých sítích se skupinovým vysíláním. Monitorování pomáhá příjemci detekovat ztrátu paketů a provést kompenzaci kolísání zpoždění v síti. RTCP používá UDP port o jedničku vyšší než používá RTP.
RTCP vytváří zpětnou vazbu mezi účastníky relace protokolu RTP, ve které periodicky probíhá výměna RTCP paketů. RTCP pakety obsahují informace, podle kterých může strana vysílající multimediální proud dynamicky měnit např. rychlost přenosu na základě požadavků strany přijímající. Protokol RTCP tak poskytuje služby řízení toku a kontroly zahlcení sítě.

RTSP

The **Real Time Streaming Protocol** (**RTSP**), developed by the IETF and published in 1998 as RFC 2326, is a protocol for use in streaming media systems which allows a client to remotely control a streaming media server, issuing VCR-like commands such as "play" and "pause", and allowing time-based access to files on a server.

Some RTSP servers use RTP as the transport protocol for the actual audio/video data. Many RTSP servers use RealNetworks's proprietary RDT as the transport protocol.

RTSP commands

RTSP requests are based on HTTP requests. While HTTP is stateless, RTSP is a stateful protocol. A session ID is used to keep track of sessions when needed, this way, no permanent TCP connection is needed. RTSP messages are sent from client to server, although some exceptions exist, where the server will send to the client. Below are the basic RTSP requests. A number of typical HTTP requests, like an OPTION request are also frequently used.

DESCRIBE

A DESCRIBE request includes an RTSP URL (rtsp://...), and the type of reply data that can be handled.

The reply includes the presentation description, typically in SDP format. Among other things, the presentation description lists the media streams controlled with the aggregate URL. In the typical case, there is one media stream for audio and one for video.

SETUP

A SETUP request specifies how a single media stream must be transported. This must be done before a PLAY request is sent.

The request contains the media stream URL and a transport specifier. This specifier typically includes a local port for receiving RTP data (audio or video), and another for RTCP data (meta information).

The server reply usually confirms the chosen parameters, and fills in the missing parts, such as the server's chosen ports. Each media stream must be configured using SETUP before an aggregate play request may be sent.

PLAY

A PLAY request will cause one or all media streams to be played. Play requests can be stacked, by sending multiple PLAY requests.

The URL may be the aggregate URL (to play all media streams), or a single media stream URL (to play only that stream). A range can be specified. If no range is specified, the stream is played from the beginning and plays to the end, or, if the stream is paused, it is resumed at the point it was paused.

PAUSE

A PAUSE request temporarily halts one or all media streams, so it can later be resumed with a PLAY request.

The request contains an aggregate or media stream URL. When to pause can be specified with a range parameter. The range parameter can be left out to pause immediately.

RECORD

The RECORD request can be used to send a stream to the server for storage.

TEARDOWN

A TEARDOWN request is used to terminate the session. It stops all media streams and frees all session related data on the server.

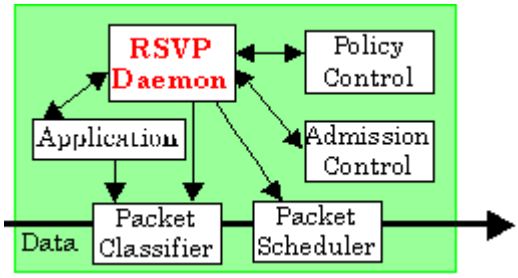
MAC header |IP header |TCP/UDP header | RTSP message

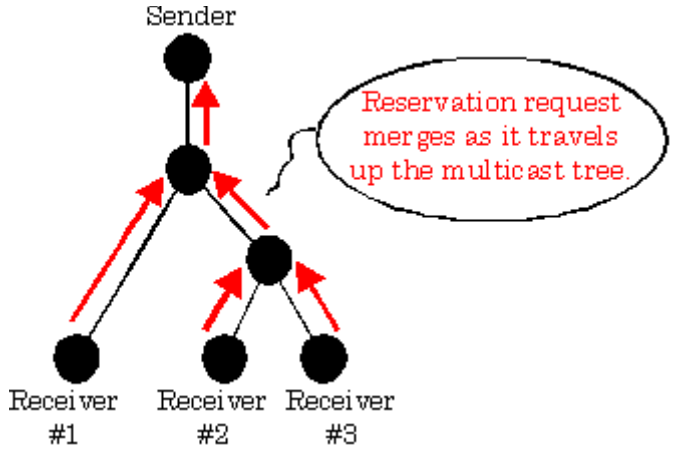
RSVP

A host uses RSVP to request a specific *Quality of Service* (QoS) from the network, on behalf of an application data stream. RSVP carries the request through the network, visiting each node the network uses to carry the stream. At each node, RSVP attempts to make a resource reservation for the stream.

To make a resource reservation at a node, the RSVP daemon communicates with two local decision modules, *admission control* and *policy control*. Admission control determines whether the node has sufficient available resources to supply the requested QoS. Policy control determines whether the user has administrative permission to make the reservation. If either check fails, the RSVP program returns an error notification to the application process that originated the request. If both checks succeed, the RSVP daemon sets parameters in a *packet classifier* and *packet scheduler* to obtain the desired QoS. The packet classifier determines the QoS class for each packet and the scheduler orders packet transmission to achieve the promised QoS for each stream.

A primary feature of RSVP is its scalability. RSVP scales to very large multicast groups because it uses receiver-oriented reservation requests that merge as they progress up the multicast tree. The reservation for a single receiver does not need to travel to the source of a multicast tree;





rather it travels only until it reaches a reserved branch of the tree. While the RSVP protocol is designed specifically for multicast applications, it may also make unicast reservations.

RSVP is also designed to utilize the robustness of current Internet routing algorithms. RSVP does not perform its own routing; instead it uses underlying routing protocols to determine where it should carry reservation requests. As routing changes paths to adapt to topology changes, RSVP adapts its reservation to the new paths wherever reservations are in place. This modularity does not rule out RSVP from using other routing services. Current research within the RSVP project is focusing on designing RSVP to use routing services that provide alternate paths and fixed paths.

RSVP runs over IP, both IPv4 and IPv6. Among RSVP's other features, it provides opaque transport of traffic control and policy control messages, and provides transparent operation through non-supporting regions.

For more information, see the RSVP publication list.

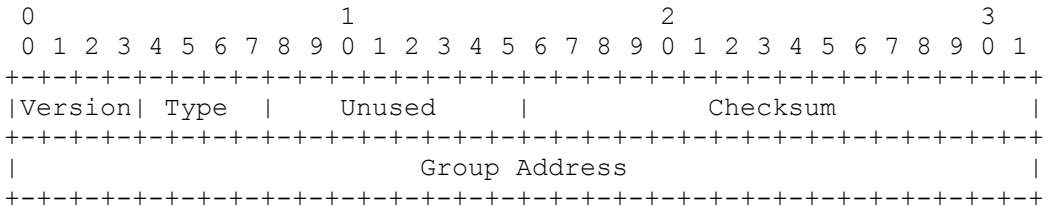
Internet Group Management Protokol

Základem pro IGMP byl Host Membership Protokol, který navrhl Steve Deering ve své doktorské práci. Verzi nula popisuje RFC-988, kdysi hojně používanou verzi 1 Deering popsal v RFC-1112, další pokračování je v RFC-2236 a zatím poslední verzi 3 popisuje v RFC-3376. Tento protokol především používají multicastoví klienti, aby signalizovali routerům své členství v multicastových skupinách. Nicméně není to jen jediné jeho využití, používá ho například i protokol DVMRP pro přenos svých zpráv a pod.

IGMPv1

Možná se zdá být trochu zbytečné popisovat protokol verze 1, který už je poměrně zastaralý, ale bohužel ještě existuje měřitelná skupina (naštěstí se již rychle zmenšující) operačních systémů, které verzi 1 používají. Je tedy nutné tuto verzi znát a rozumět jejím omezením. Ještě mnohem silněji to pochopitelně platí u verze 2.

Formát IGMP zprávy je poměrně jednoduchý. Po IP hlavičce následuje tato sekvence osmi bytů:



Význam jednotlivých políček je následující:

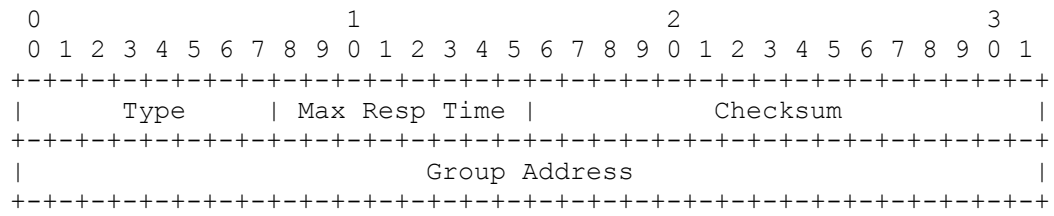
- Version - aktuální verze protokolu (1),
- Type - tato verze podporuje pouze dva typy zpráv:
  - Membership Query, kód 1,
  - Membership Report, kód 2,
- Unused - nepoužito, při posílání by mělo být nastaveno na 0 a při příjmu ignorováno,
- Checksum - kontrolní součet IGMP zprávy,
- Group Address - u zprávy Membership Report obsahuje toto pole multicastovou adresu, jinak by mělo být nastaveno na 0.0.0.0.

Protokol má jednoduché schéma Query-Report. Pokud se nějaký klient chce připojit k multicastové skupině a pošle zprávu typu Membership Report s vyplněným číslem skupiny na adresu té skupiny, příslušný router by ji měl zachytit a postarat se o zprostředkování. Router si průběžně kontroluje členství klientů ve skupinách a občas (jednou za 60 sekund) pošle zprávy typu Membership Query na adresu 224.0.0.1 a čeká na Reporty klientů. Protože je běžné, že klientů pro jednu skupinu je v síti více, byl navržen mechanismus, který zaručí, že pro každou skupinu dorazí pouze jediný Report. Každý z klientů, který uslyší Query, si zvolí náhodné číslo v rozsahu 0-10. Toto číslo vyjadřuje čas v sekundách, za který chce poslat Report. Zároveň ale poslouchá, zda nějaký jeho kolega Report pro danou skupinu pošle. Pokud se tak stane, klient už nic neposílá.

V případě, že klient danou skupinu opustí, přestane jednoduše odpovídat na Query. Pokud router nedostane pro skupinu třikrát po sobě žádný Report, přestane ji posílat. Query se posílají obvykle jednou za minutu - v nejhorším případě tedy router posílá data ještě tři minuty po té, co už o ně nikdo nestojí. Tento jednoduchý mechanismus je velmi nevýhodný, pokud klient často mění členství ve skupinách. V takovém případě může sít snadno přetížit.

IGMPv2

Zprávy IGMPv2 vypadají jakoby nekompatibilně s předchozí verzí. Obsahují trochu jiná políčka:



Jejich význam je následujíci:

- Type - hodnoty totoho pole jsou voleny tak, aby bylo možné rozeznat zprávy IGMPv1, které na tomto místě mají políčka dvě - version a type. Protokol rozeznává následující typy zpráv:
  - Membership Query - vlastně jsou dvě, General Query a Group-specific Query, rozlišení se dělá podle obsahu políčka Group Address, číselný kód zprávy je 11, tedy zpráva vypadá podobně jako IGMPv1 Query,
  - IGMPv1 Membership Report - pro zpětnou kompatibilitu, kód zprávy 12,
  - IGMPv2 Membership Report, kód 16,
  - Leave Group, kód 17,
- Max Resp Time - používá se u Query zprávy a označuje maximální čas na zaslání reportu v desetinách sekundy. Mechanismus je stejný jako u IGMPv1,
- Checksum - kontrolní součet IGMP zprávy,
- Group Address - u zpráv Membership Report, Leave Group a Group-specific Query obsahuje toto pole multicastovou adresu, jinak by mělo být 0.0.0.0. Z toho je vidět, že zpráva General Query u IGMPv2 je až na pole Max Resp Time shodná s Membership Query u IGMPv1.

Signalizace přihlášení do skupiny je obdobná jako u IGMPv1. Novinkou je proces opuštění skupiny. Klient může poslat zprávu (a obvykle posílá) Leave Group. Router na takovou zprávu zareaguje posláním Group-specific Query do dané skupiny. Max Resp time je obvykle nastaven na 1 sekundu. Pokud je ještě někdo členem skupiny, odpoví, a router pokračuje v posílání dat.

Další novinkou je volba routeru, ktery posílá Query. Je-li na síti více routerů, je pro posílání Query vybrán ten s nejvyšším IP. Ostatní pouze poslouchají a jsou připraveni převzít jeho roli. Mechanismus pracuje tak, že pokud router uslyší Query zprávu od routeru s vyšším IP, přestane ty své sám posílat a čeká 400 sekund, zda nepřijde další. Pokud nepřijde, vyhodnotí, že vybraný server už asi nefunguje, a začne posílat Query sám, čímž proběhnou nové volby. V IGMPv2 se Query posílá jednou za 125 sekund.

IGMPv2 je zpětně kompatibilní s IGMPv1. Pokud klienti detekují IGMPv1 router dle jeho Query, začnou posílat zprávy také v IGMPv1 a nastaví si časovač na 400 sekund. Pokud nastavený interval uplyne, začnou opět posílat IGMPv2. Naopak server je schopen obsloužit mix IGMPv1 i IGMPv2 klientů, protože umí jejich zprávy rozlišit. IGMPv1 klienti nepoznají žadný rozdíl. Pokud router detekuje IGMPv1 klienty na síti, ignoruje Leave Group zprávy.

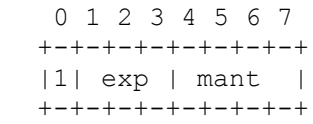
IGMPv3

Specifikace IGMPv3 dokázala zatím ještě zhruba přehlednou situaci výrazně zkomplikovat. Problém, který se tato verze snaží řešit, je v tom, že pokud jste členem nějaké multicastové skupiny, nechcete často přijímat zprávy od všech, ale pouze od vybraných zdrojů. Proto se v IGMPv3 nepřihlašujete pouze do skupiny (*, G), ale přihlašujete se k odběru dat z konkrétního zdroje v dané skupině (S, G). Přirozeně se výrazně změnil i formát zpráv. Zprávy již nemají konstantní délku, jako tomu bylo u předchozích verzí. Přesný popis formátu by sám zabral nejméně jeden článek, takže jej vypustíme. IGMPv3 má své dvě vlastní zprávy a pro zpětnou kompatibilitu rozumí dalším třem:

- Membership Query - kód 11,
- IGMPv3 Membership Report - kód 22,
- IGMPv1 Membership Report - kód 12,
- IGMPv2 Membership Report - kód 16,

- IGMPv2 Leave Group - kód 17.

Význam zpráv je stejný jako u předchozích verzí, pouze se přidávají políčka pro členství ve skupinách. Také se trochu změnil význam políčka Max Resp Time. Nyní se jmenuje Max Resp Code a pokud je jeho hodnota nižší než 128, význam se nemění. Pokud se ale používá vyšší hodnota (první bit je 1), změní se význam na:




```
Max Resp Time = (mant | 0x10) << (exp + 3)
```

IGMPv3 Report se také neposílá do příslušné skupiny, ale na adresu 224.0.0.22.

Prvních 8 bytů Query zprávy je stejných jako u předchozích verzích, takže klienti s nižší verzí protokolu nepoznají rozdíl a odpoví zprávou Report v odpovídající verzi. Router pak zachází s každou skupinou dle verze přihlášených klientů. Pokud se tedy do stejné skupiny na stejné síti přihlásí IGMPv2 i IGMPv3 klient, router nebude provádět filtrování zdrojových adres a bude posílat všechna data skupiny. Naopak pokud klient dostane zprávu délky 8 bytů, ví, že jeho router používá IGMP nižší verze, a přizpůsobí se mu.

TFPT

V čem se liší TFTP od ftp

Protokol TFTP (Trivial File Transfer Protocol) se od protokolu FTP liší především v následujících aspektech:

- pro transport dat používá služeb protokolu UDP (zatímco FTP používá transportní služby protokolu TCP)
- nezajišťuje žádné systémové akce na vzdáleném počítači (typu výpisu adresáře, změny aktuálního adresáře, rušení souborů apod.),
- nezajišťuje žádnou identifikaci uživatele, který žádá o přenos
- přenáší jen textové a binární soubory.

Soubory, které přenáší, chápe protokol TFTP buď jako textové soubory, nebo jako soubory binární. V prvním případě předpokládá (obdobně jako protokol FTP), že jednotlivé znaky jsou při přenosu kódovány přesně takovým způsobem, jaký požaduje protokol Telnet (a příjemce či odesilatel je v případě potřeby konvertuje z/do místních konvencí), zatímco v druhém případě se na obsah přenášeného souboru dívá jako na posloupnost osmibitových bytů, a nijak je neinterpretuje.

Pro vlastní přenos využívá protokol TFTP nespolehlivých a nespojovaných služeb transportního protokolu UDP. S jeho nespolehlivostí se vyrovnává tak, že si sám zajišťuje potřebné potvrzování. Toto potvrzování je zajímavé tím, že jde o tzv. jednotlivé potvrzování (tedy takové, kdy odesilatel po odeslání jednoho bloku čeká na jeho explicitní potvrzení, a teprve pak vysílá další blok), a dále tím, že používá symetrické čekání na vypršení časového limitu (tzv. timeout). Odesilatel, který odeslal nějaká data, čeká na jejich potvrzení, a pokud jej nedostane do určitého časového limitu, vyšle data znovu. Obdobně příjemce, který data obdrží, je potvrdí odesláním příslušného potvrzení, ale pokud do časového limitu nedostane další data, znovu vyšle již jednou odeslané potvrzení. Jednotlivé bloky dat, které jsou tímto způsobem přenášeny, mají pevnou velikost (512 bytů), a jsou sekvenčně číslovány od 1. Podobně jsou sekvenčně číslována i potvrzení. Konec celého přenosu příjemce rozpoznává podle posledního bloku, který musí obsahovat méně než standardních 512 bytů (tedy např. i 0 bytů).

V prvním (resp. nultém) bloku, kterým se přenos zahajuje, musí být uvedeno přesné jméno souboru, který má být přenesen, včetně úplné přístupové cesty k tomuto souboru. Protokol TFTP, na rozdíl od protokolu FTP, totiž nepočítá s tím, že by na straně serveru byl nastaven na nějaký konkrétní adresář. V důsledku toho ani nezná pojem aktuálního adresáře (na serveru), neumožňuje přecházet mezi jednotlivými adresáři, a nezprostředkovává ani jejich výpis. Počítá s tím, že klient přesně ví, kde a s jakým souborem chce pracovat.

Jménem koho?

Velmi zajímavé důsledky vyplývají z dalšího zjednodušení protokolu TFTP. Na rozdíl od "plnohodnotného" protokolu FTP totiž nezná pojem uživatele. Nepočítá s tím, že by uživatel v roli klienta prokazoval serveru svou totožnost (resp. toto neumožňuje), a tak všechny jeho požadavky na čtení či zápis jednotlivých souborů jsou vlastně anonymní. Jak má ale server posuzovat jejich oprávněnost či neoprávněnost?

Definice protokolu TFTP toto ponechává na implementaci, ale obvyklé řešení je takové, že číst lze jen ty soubory, které jsou ke čtení přístupné všem uživatelům. V případě zápisu pak rozhodují přístupová práva do konkrétního adresáře (zda umožňuje zápis všem uživatelům či nikoli), resp. přístupová práva ke konkrétnímu souboru (jde-li o přepis nebo o přidávání za konec již existujícího souboru).

Anonymní FTP servery

Na problém s přístupovými právy narazíme i v okamžiku, kdy se rozhodneme vytvořit na nějakém počítači veřejně přístupný archiv, ze kterého by si kdokoli mohl nahrávat zde umístěné soubory. Jaký protokol pro přenos souborů je k tomuto účelu nejvhodnější?

Protokol TFTP by byl vhodný právě pro svou "anonymitu", díky které by zájemci o přístup k archivu nemuseli uvádět žádná uživatelská jména ani hesla. Má to ovšem jeden malý háček - jelikož protokol TFTP neumožňuje procházet adresáři serveru a vypisovat si jejich obsah, nemohl by si zájemce sám vyhledávat to, co jej zajímá či co potřebuje, ale musel by být s obsahem archivu seznámen jiným způsobem, a pak jít "na jistotu".

Proto se ke zpřístupnění nejrůznějších archivů souborů v sítích na bázi TCP/IP používá "plnohodnotný" protokol FTP. I zde je ale malý háček - má-li být archiv opravdu veřejně přístupný, jak sdělit všem potenciálním zájemcům potřebné uživatelské jméno (a případně i heslo), které mají použít? Možným řešením je zvolit jedno konkrétní jméno, a to pak používat všude. Tedy vlastně zavést jednotnou konvenci, a tu důsledně dodržovat. No a jaká že tato konvence je? V síti Internet, která je dnes zdaleka nejvýznamnější sítí na bázi protokolů TCP/IP a pokrývá prakticky celý svět, je touto jednotnou konvencí uživatelské jméno **anonymous** (doslova: anonym). Podle něj jsou pak také příslušné veřejně přístupné archivy označovány jako **anonymní FTP servery** (**anonymous FTP servers**). V celém Internetu jich je opravdu mnoho, jen jejich stručné seznamy mívají desítky stránek.

Pokud se na některý z nich obrátíte (v roli klienta protokolu FTP), můžete se přihlásit jako

uživatel "anonymous". Heslo pak na vás již vůbec není požadováno, nebo jste vyzváni k tomu, aby jste místo hesla uvedli svou skutečnou identitu (míněno: svou adresu pro elektronickou poštu) - viz též obrázek 72.3 v minulém dílu seriálu. Některé anonymní FTP servery používají takto získanou informaci jen pro vlastní evidenci (aby měly přehled, kdo a jak je využívá). Jiné anonymní servery však mohou být méně důvěřivé, a ověřují si, zda vámi uvedená identita je skutečná, či nikoli. A pokud dojdou k závěru, že je smyšlená (nebo nejsou schopny její pravost ověřit), jednoduše se s vámi přestanou bavit.

OSPF

Protokol OSPF (Open Shortest Path First) je určen k výměně informací o směrování v rozsáhlých a velmi rozsáhlých strukturách propojených sítí. Tato funkce není k dispozici v operačních systémech Windows pro počítače s procesorem Itanium. Tato funkce není k dispozici v operačních systémech Windows pro počítače s procesorem řady x64.

Největší výhodou protokolu OSPF je jeho efektivita - tento protokol ani ve velmi rozsáhlých sítích nepředstavuje nijak významné zatížení přenosových cest. Jeho největší nevýhodou je složitost - jeho použití je třeba pečlivě naplánovat a také nastavení konfigurace a správa jsou komplikovanější.

Protokol OSPF počítá trasy ukládané do směrovací tabulky pomocí algoritmu SPF (Shortest Path First). Tento algoritmus hledá nejkratší (nejméně nákladnou) cestu mezi směrovačem a jednotlivými dostupnými sítěmi. Trasy vypočítané algoritmem SPF nikdy neobsahují uzavřené smyčky.

Směrovače OSPF si nevyměňují jednotlivé položky směrovacích tabulek jako směrovače RIP, ale udržují mapu struktury propojených sítí a aktualizují ji při každé změně síťové topologie. Tato mapa, nazývaná databáze stavu linky, je synchronizována mezi všemi směrovači a slouží k výpočtu tras ukládaných do směrovací tabulky. Mezi sousedícími směrovači OSPF vzniká logický vztah sloužící k synchronizaci databáze stavu linky.

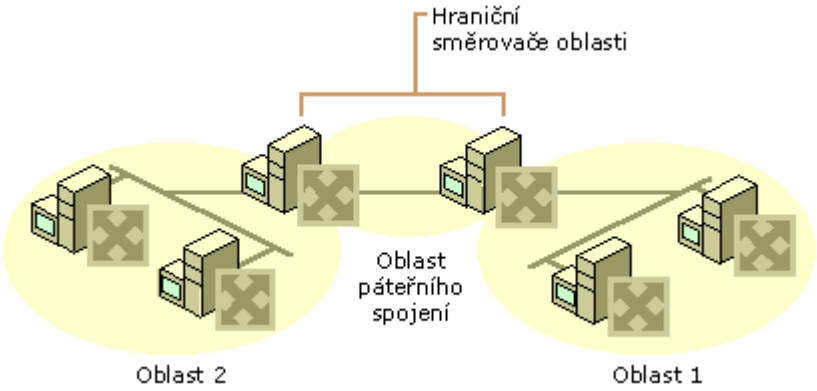
Změny topologie propojených sítí jsou efektivně šířeny v celé síťové struktuře tak, aby databáze stavu linky na jednotlivých směrovačích byly neustále shodné a aktuální. Při příjmu změn databáze stavu linky dochází k přepočtu směrovací tabulky.

Se zvětšováním databáze stavu linky rostou nároky na paměť a na dobu přepočtu tras. Protokol OSPF tento problém řeší rozdělením struktury propojených sítí na oblasti (skupiny sousedících sítí), které jsou navzájem propojeny páteřní oblastí. Databáze stavu linky jednotlivých směrovačů obsahují pouze údaje o oblastech, ke kterým je daný směrovač připojen. Spojení páteřní oblasti s ostatními oblastmi zajišťují směrovače ABR (Area Border Router).

Z důvodu dalšího omezení množství informací šířených do jednotlivých oblastí umožňuje protokol OSPF použití oblastí se zakázaným inzerováním. Oblast se zakázaným inzerováním může obsahovat jediný vstupní a

výstupní bod (jediný směrovač ABR) či více směrovačů ABR, kdy každý ze směrovačů ABR je možné použít k dosažení externích tras k cílům.

Na následujícím obrázku je zobrazeno schéma síťové struktury propojené pomocí protokolu OSPF.



Protokol OSPF má oproti protokolu RIP následující výhody:

- Trasy vypočítané pomocí protokolu OSPF nikdy neobsahují uzavřené smyčky.
- Protokol OSPF se dokáže přizpůsobit i velmi rozsáhlým síťovým strukturám.
- Úprava konfigurace v případě změny síťové topologie je rychlejší.

Implementace protokolu OSPF ve službě Směrování a vzdálený přístup nabízí následující funkce:

- směrovací filtry zajišťující řízení komunikace s jinými směrovacími protokoly,
- dynamické změny konfigurace všech parametrů protokolu OSPF,
- možnost koexistence s protokolem RIP,
- dynamické přidávání a odstraňování rozhraní.

FIREWALLY

Firewall pracuje ve dvou základních metodách, ale v každé z nich nějakým způsobem vyhodnocuje právě ty přenášené pakety. Buď vyhodnocuje IP datagram a to zda vyhovuje nastaveným podmínkám - pak se jedná o paketový filtr a říká se, že se jde o síťovou vrstvu a nebo se zabývá obsahem jednotlivých paketů - jde o tzv. aplikační bránu a tudíž se hovoří o aplikační vrstvě.

Avšak není výjimkou, že firewall pro svou spolehlivou funkci využívá obě dvě zmíněné možnosti zároveň. Rozeberu zde obě možnosti postupně.

Paketové filtry

Nejčastěji se paketové filtry vyskytují v systémech UNIX, jež je mají přímo zabudované v jádrech. V systémech Windows se vyskytují pouze ve verzích 2000 a XP, avšak nedosahují takových kvalit jako právě v systémech UNIX.

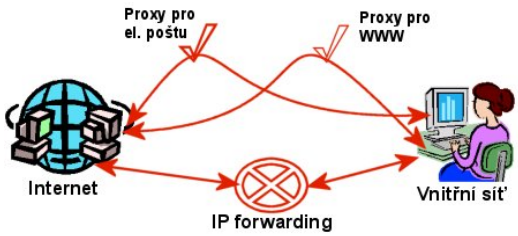
Názorné schéma kontroly paketů.

Pracují tak, že vyhodnocují jednotlivé pakety podle hlavičky IP datagramu, jež obsahuje IP adresu odesílatele i příjemce, zdrojový i cílový port a další informace. Paketový filtr má daná pravidla, podle kterých pak ze získaných informací z IP datagramu rozhoduje, zda paket projde a nebo bude zahozen. Pracují však pouze na úrovni uvedených vlastností paketů a nedokáží vyhodnotit jejich obsah. To znamená, že nedokáží například rozpoznat autorizovaného uživatele. Samotné paketové filtry jsou velice rychlé a nenáročné na systémové zdroje. Hlavní výhodou je, že nepotřebuje žádnou podpůrnou aplikaci a celý proces se odehrává přímo v jádře. V dnešní době se paketové filtry vyskytují už ve všech nových operačních systémech jako jsou UNIX, BSD, Windows XP, Mac OS a i Linux. Pokud někdo řekne, že používá firewall, tak v 80% se jedná právě o paketový filtr.

Aplikační brány

Někdy se můžete setkat s označením jako aplikační gateway, Proxy, gateway, aplikační Proxy. Je to vlastně software běžící na firewallu. Existují dva nepatrně odlišné způsoby fungování.

První z nich je podobný paketovému filtru jen s tím rozdílem, že rozhodování se odehrává s využitím informací jednotlivých aplikačních vrstev (kontroluje obsah paketu). Chová se vlastně navenek jako směrovač, neboť pracuje pouze v síťové vrstvě, avšak interně zasahuje až na aplikační vrstvu. Což znamená, že nevyhodnocuje jen odkud kam paket jde, ale sleduje i jeho obsah.



Častěji se však používá řešení druhé. Celý proces komunikace uživatele s internetem skrze proxy gateway probíhá poněkud složitěji. Pokud uživatel zadá do svého internetového prohlížeče (browseru) nějakou adresu, tak ta by měla dojít k příslušnému serveru, tam se zpracovat a zpět odeslat internetovou stránku. Avšak při použití proxy (je to vlastně server a klient v jednom společném provedení, který se stará o jednu příslušnou službu jako například FTP, WWW, e-mail...) nemůže uživatel vznést svůj požadavek přímo na server, ale musí jej odeslat skrze prostředníka jimž je právě proxy. Brána požadavek příjme, sama vygeneruje požadavek svým jménem a odešle jej příslušnému serveru. Když pak příjme zpět výsledek, tak jej pošle zájemci.

Hlavní výhodou firewallu používajícího proxy je ta, že již nemůže zůstat pro koncového uživatele transparentní. To znamená, že při kliknutí na odkaz se informace neodešle přímo na cílový server, ale na proxy bránu, která teprve zajistí vše, co je třeba. Jelikož tedy data procházejí skrze proxy gateway, je možné je zaznamenávat, monitorovat či různě blokovat.

Jednou z mnoha dalších funkcí, které může proxy brána plnit, je funkce cache. To znamená, že si uchovává již jednou navštívené WWW stránky, aby při dalším požadavku na jejich zobrazení je nebylo nutné stahovat z internetu a tím zatěžovat linku, ale stačilo je pouze poslat ze své vnitřní cache paměti.

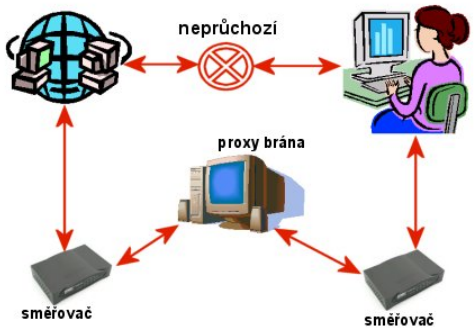
Omezení či povolení

Ať už zvolíte firewall kterýkoli, vždy je hlavně potřeba nastavit firewall pro jeden z těchto dvou způsobů povolování či zakazování paketů. Buď můžete nastavit implicitní zákaz - jedná se o to, že zakážete všechny služby sítě a povolíte jen ty, které pro provoz potřebujete. Druhá možnost je přesně opačná - implicitní povolení. Při této volbě povolujete všechny služby a zakazujete jen ty, které považujete za rizikové. Ve výsledku má každá z těchto dvou možností své klady a zápory a jde především o to, pro jaký typ sítě se která hodí více. Avšak u obou je potřeba při zavádění důkladně dbát na to, aby nebyla žádná z možných služeb opomenuta, neboť právě ta jedna by mohla útočníkovi poskytnou volnou cestu.

Architektura firewall

V praxi se celková architektura firewallu dělí na dva základní prvky jimiž jsou demilitarizovaná zóna a dvounohé firewally. Jejich popis, výhody a nevýhody zde rozepíšu zvlášť.

Demilitarizovaná zóna



Při použití tohoto zapojení je veškerá komunikace přenášena skrze směrovače a samotný firewall. Není možné, aby PC odeslalo svůj požadavek přímo na server do Internetu (je zakázán IP-forwarding). Demilitarizovaná zóna je tedy oblast od jednoho směrovače po druhý.

Přenos packetů tedy probíhá následovně. Z PC je odeslán požadavek na Internetový server. Požadavek jde přes směrovače do proxy brány, kde se vyhodnotí a jménem brány odešle příslušnému serveru. Ten jej po obdržení zpracuje a zpět odešle požadovanou informaci. Ta dojde zpět na proxy bránu, zkontroluje se a buď se zahodí, nebo se pošle na PC, odkud požadavek vyšel.

Při tomto zapojení je stupeň bezpečnosti na nejvyšší úrovni, neboť potencionální útočník by musel projít přes dva síťové směrovače. Nevýhoda tohoto zapojení je ta, že je zapotřebí použít dva směrovače, jež musí umožňovat posílání packetů pouze v rámci demilitarizované zóny a packety, jež by měly tuto hranici přesahovat musí zahodit. Další nevýhodou je větší cena tohoto zapojení právě z důvodu použití dvou směrovačů.

Další z možností je zapojení pomocí jednoho směrovače, který ovšem musí mít minimálně tři síťové rozhraní. Samozřejmě tyto postupy nejsou jen ty jediné možné. Existuje celá řada různých funkčních zapojení, které se volí podle typu a požadavků.

Jedním z velice často používaných zapojení je pomocí samostatného firewallu jež plní jak funkci mostu a směrovače, tak i již zmíněnou funkci firewallu. Toto zapojení se používá hlavně v domácnostech a nebo v menších firmách, kde počet PC není příliš vysoký. Výhodou však je, že není třeba dokupovat směrovače.

Dvounohé firewally

Tento způsob zapojení je prakticky ukázán na předchozím obrázku. Jedná se vlastně o směrovač, jež má navíc za úkol pracovat jako firewall se všemi svými možnostmi. Je potřeba, aby dvounohý firewall měl alespoň dvě síťová rozhraní, neboť jedno slouží k připojení veřejné sítě (v tomto případě internetu) a k druhému se napojí místní síť LAN. Opět umožňuje pouze komunikaci skrze proxy a jakýkoliv ip-forwarding je zakázán. Může plnit funkci aplikační brány a zároveň paketového filtru.

RIP

RIP protokol se používá již od doby počátků ARPANETu a je založený na protokolu firmy Xerox, používaném v 70. létech. V současné době existuje verze RIP-2, která rozšiřuje možnosti původního RIP-1 protokolu. Výhodou protokolu je jednoduchost, nesoucí s sebou i snadnou implementovatelnost. Má ale své podstatné nevýhody:

- omezení šířky sítě (nejdelší cesty) na 15. Metrika 16 je používána jano nekonečno, takže cesta takové délky je nedosažitelná.
- přenášení relativně značného množství informací může zahlcovat síť - směrovače posílají své směrovací informace okolním směrovačům každých 30 sec
- nedostatečná podpora podsítí
- nedostatečné zabezpečení

Třetí a čtvrtý problém je řešen v RIP-2.

RIP směrování je založené na Bellman-Fordově algoritmu, označovaném rovněž jako distance vector algoritmus. Každý směrovač si udržuje tabulku s minimálně následujícími informacemi pro každý vstup:

- IP adresa cílového uzlu - počítače nebo sítě
- IP adresa prvního směrovače, vedoucímu k cílovému uzlu
- metrika - vzdálenost k cílovému uzlu
- časovače
- flag - určující, zda byl záznam změněn

Metrikou se míní součet metrik segmentů jednotlivé cesty. Obvykle jsou všechny nastaveny na jedna, ale v případě potřeby je možné nastavit vyšší hodnoty a určit tak preference jednotlivých cest.

Na počátku zná směrovač pouze vzdálenosti k okolním směrovačům. Ty jsou dány metrikou sítě. Vzdálenost k sobě samému se nastaví na jedna nebo může určovat metriku sítě přímo přes interface připojené k danému směrovači. Směrovač pak v pravidelných intervalech (standardně 30 s) posílá své směrovací informace do okolních směrovačů, čímž se informace šíří dál a dál v síti. Hodnoty pro konkrétní cílovou IP adresu jsou aktualizovány (novými hodnotami získanými z okolního uzlu) v případě, že metrika je nižší nebo pokud je nová hodnota získaná ze stejného uzlu jako původní hodnota. Nová hodnota je dána součtem získané hodnoty a vzdálenosti od uzlu, ze kterého přišla. V případě, že některý ze směrovačů přestane pracovat, směrovač, který tuto skutečnost zjistí, aktualizuje příslušný záznam ve směrovací tabulce na nekonečno, které tak označuje nedostupnost uzlu. Nefunkčnost směrovače je zjištěna v případě, že vyprší hodnota časovače, asociovaném s danou cestou.

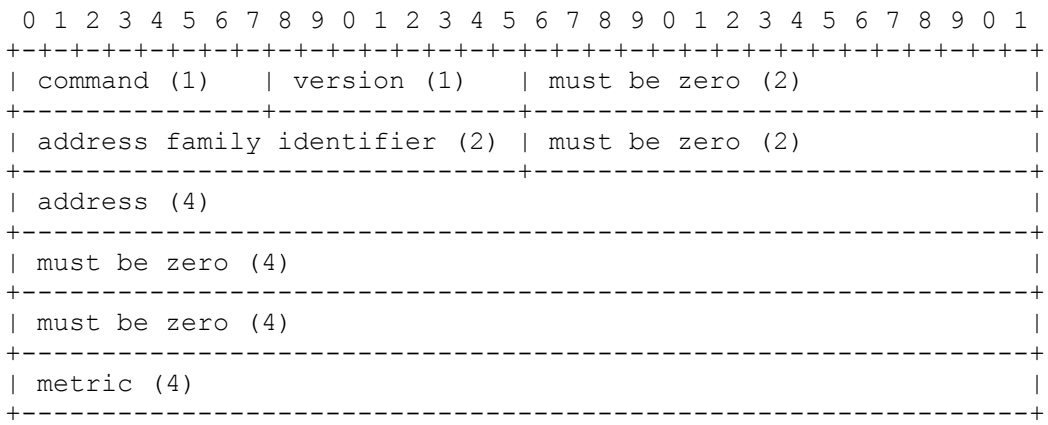
Takový algoritmus by byl sice funkční, ale jen velmi pomalu by reagoval na změny topologie. Řešením je nečekat vždy do doby, kdy se posílají pravidelné aktualizace, ale vyslat aktualizaci vždy, pokud je směrovací tabulka změněna. Takové aktualizační zprávy se nazývají **triggered updates** a výrazně zrychlí šíření směrovacích informací.

Problém nastane v případě, kdy např. uzly A a B si myslí, že se mohou dostat do jiného uzlu (např. C) vždy přes toho druhého, tzn. A přes B a B přes A. Situace evidentně vede ke vzniku cyklu. Řešením je tzv. rozdělěný horizont (**split horizon**). Myšlenka je taková, že směrovač nebude posílat záznam ze své směrovací tabulky do uzlu, ze kterého tento záznam získal. Modifikací je **split horizon with poisoned reverse** - směrovač pošle záznam i do směrovače, ze kterého informaci zjistil, ale s metrikou nekonečno, čímž dojde k zamezení použití této cesty. Toto řešení je bezpečnější, ale vede k větší zátěži sítě.

Každý uzel vlastní několik časovačů. Jeden z nich je společný pro všechny záznamy v tabulce a používá se pro měření intervalů, po kterých se posílají aktualizační zprávy. Standardní hodnota je 30 s. Další dva časovače jsou asociovány s jednotlivými záznamy ve směrovací tablulce. **Timeout** časovač odměřuje dobu od poslední aktualizace odpovídajícího záznamu a standardní hodnota je 180 s. Po uplynutí tohoto časového úseku se má za

to, že cesta je nedostupná a začne proces rušení. Ten spočívá v nastavení metriky odpovídajícího záznamu na nekonečno a rozběhnutí **garbage-collection** časovače. Ten je nastaven na 120 s a odměřuje dobu, po kterou záznam ještě existuje v tabulce a je tedy posílán do okolních uzlů. Po vypršení garbage-collection časovače je záznam ze směrovací tabulky vymazán.

RIP je založený na UDP protokolu a komunikace probíhá na portu 520.
Následuje formát RIP-1 paketu: (čísla v závorkách udávají délku v bajtech)



Část mezi adress family identifier a metrikou se může opakovat až 25x.

Command může být:

- 1 - **request** - požadavek na zaslání části nebo celé tabulky
- 2 - **response** - část nebo celá směrovací tabulka - může být jako odpověď na request nebo pravidelně zaslaná aktualizace (update)

Hodnoty 3 a 4 (traceon, traceoff) jsou zastaralé a hodnota 5 je rezervovaná Sun Microsystems pro vlastní účely.

Verze určuje verzi RIP protokolu, tedy 1 nebo 2.

Adress family identifier určuje typ adresy. Podporovány jsou v současnosti pouze IP adresy a adress family identifier pro ně má hodnotu 2.

Adresou se rozumí jedno z následujících:

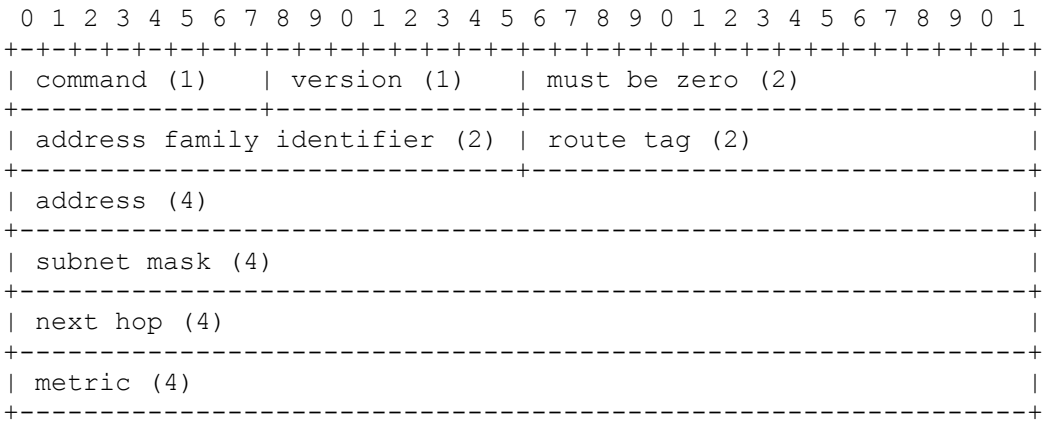
- host adress - adresa cílového počítače
- subnet number
- network number
- 0.0.0.0 - označení default cesty

Směrování podle čísla podsítě je proveditelné jen za předpokladu znalosti masky podsítě. Běžně jsou známy pouze masky podsítě přímo připojených sítí. Je třeba zavést filtrování směrovacích informací v hraničních směrovačích (border). Takový směrovač je jednak připojen přímo k podsítím a jednak k dalším směrovačům nepřímo. Do směrovačů přímo připojené podsítě se posílají čísla podsítě, naopak do ostatních směrovačů se pošle jediné číslo celé sítě. Distribuování čísel podsítě by nemělo smysl, protože okolní směrovače neznají masky daných podsítí a nebyly by tak schopny korektně směrovat.

Default cesta, označená adresou 0.0.0.0, je použita v případě, že cílová adresa neodpovídá žádnému záznamu ve směrovací tabulce. Může se jednat např. o cestu ke směrovači, spojující autonomní systém s okolními systémy (přes EGP). Podpora adresy 0.0.0.0 není v implementacích vyžadována, ale silně se doporučuje. Pokud podporována není, pak všechny vstupy s neodpovídající adresou musí být ignorovány.

RIP verze 2

Formát RIP-2 paketu vypadá následovně:



Číslo verze je pro RIP-2 nastaveno na 2.

Route tag má umožnit spolupráci na úrovni IGP-IGP nebo IGP-EGP. Pro cesty, získané z vnějšku systému (tj. z EGP nebo jiného IGP) by měl být nastaven na určitou hodnotu, např. identifikující číslo autonomního systému, odkud cesta pochází.

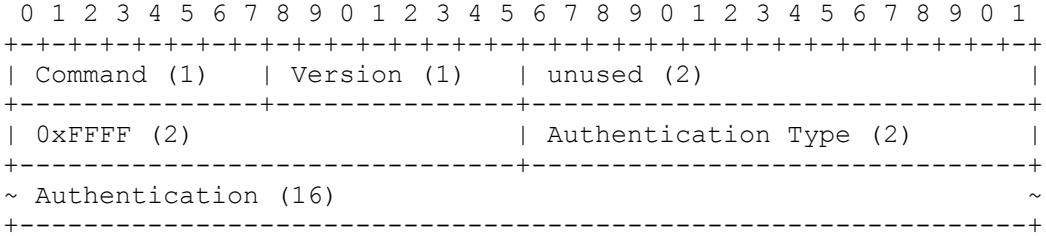
Subnet mask je maska podsítě příslušná k damému číslu podsítě. Pokud adresou není číslo podsítě, je maska nastavena na nulu.

Next hop je IP adresa, na kterou by měly být pakety pro danou cílovou adresou posílány a měla by být přímo dostupná na dané logické podsíti. Specifikování next hop má vést k zamezení zbytečnému prodlužování cest. Hodnota 0 určuje, že next hop není nastaven a použije se klasického RIP směrování. Stejně tak se postupuje v případě, že next hop není dostupný.

Ostatní zůstává stejné jako u RIP-1.

RIP-2 pro pravidelné posílání aktualizačních zpráv (update) sousedním směrovačům nepoužívá broadcasting, ale multicasting na adresu 224.0.0.9.

RIP-2 přináší ještě možnost ověřování autentičnosti (authentication). Ta je realizována zadáním adress family identifieru na 0xFFFF:



Dále pak následují záznamy ve výše uvedeném formátu (adresa....metrika). Vzhledem k tomu, že první záznam je autentikační, těch dalších (s informacemi o směrovacích cestách) může být max. 24.

V současnosti jediným podporovaným typem autentikace je **simple password** a odpovídající číslo typu je 2. V poli autentikace je pak heslo, zarovnané doleva a doplněné nulami.

NVT - Network Virtual Terminal

Virtuální terminál, používaný protokolem TELNET, je označován jako **NVT**, neboli **Network Virtual Terminal** (doslova: síťový virtuální terminál). Je zvláštním případem parametrického modelu v tom smyslu, že předpokládá pevně danou hodnotu jednotlivých parametrů, které terminál definují (viz 66. díl), čímž vlastně fixuje jeho vlastnosti i konkrétní způsob ovládání. Popišme si nyní, jaká je představa tohoto zařízení:

NVT je obousměrné, znakově orientované zařízení, které lze nejlépe přirovnat ke dvojici klávesnice-tiskárna. Klávesnice generuje jednotlivé znaky v kódu ASCII, zatímco tiskárna je průběžně tiskne. Celek pak odpovídá představě tzv. **znakového**, resp. **řádkového terminálu** (**scroll mode terminal**, viz 66. díl).

Ačkoli protokol TELNET využívá pro přenos dat spolehlivé a spojované přenosové služby transportního protokolu TCP, které vytváří plně duplexní spojení, NVT toto spojení využívá jen v poloduplexním režimu. Díky tomu je pak možné vystačit i s takovým skutečným terminálem, který je fyzicky poloduplexní (jako např. terminál IBM 2741).

NVT dále předpokládá, že přenos dat bude tzv. bufferován - tedy že data nebudou vysílána po jednotlivých znacích, ale že se budou nejprve hromadit ve vhodných vyrovnávacích pamětech (anglicky: buffers), a skutečně vysílány pak budou až větší celky. Jaké celky to ale mají být?

U řádkového terminálu, jakým je NVT, je jednoznačným kandidátem řádka. Délka řádky ovšem není pevně stanovena, a to ani u tiskárny. Na straně klienta určuje skutečnou délku každé jednotlivé řádky uživatel, který pracuje na příslušném terminálu - tím, že v určitý okamžik zmáčkne tlačítko ENTER, RETURN, NEW LINE či jak se na jeho terminálu jmenuje klávesa, kterou se zadává konec řádky. Klientská složka protokolu TELNET, která zpracovává uživatelův vstup z klávesnice, může na základě zmáčknutí této klávesy obdržet různý kód (např. jen znak CR, jen znak LF, dvojici znaků CR a LF apod.), podle konkrétního použitého terminálu. Sama však musí na jeho základě vygenerovat dvojici znaků CR a LF, neboť virtuální terminál NVT počítá s tím, že řádky budou zakončovány právě tímto způsobem. Analogicky je tomu i na straně serveru - aplikační proces, který generuje data, určená k zobrazení na terminálu, je člení na jednotlivé řádky takovým způsobem, jaký předpokládá příslušná "místní" konvence. Serverová složka protokolu TELNET je však před odesláním překládá do takového tvaru, aby byly zakončeny dvojicí CR LF.

Protokol TELNET tedy předpokládá, že data budou standardně přenášena po celých řádcích. To ale nemusí být vždy možné - není-li délka řádky předem omezena, může se stát, že pro ni nebude k dispozici dostatečně velká vyrovnávací paměť. Také to ale nemusí být vždy žádoucí - někdy může být vhodné, či dokonce nutné odesílat menší celky než celé řádky, až např. po jednotlivé znaky. S touto možností protokol TELNET počítá, a doporučuje ji realizovat. Přesný mechanismus, kterým by si zdroj dat mohl vynutit jejich odeslání ještě před zakončením řádky, je však ponechán na konkrétní implementaci.

Ostatní je na vzájemné dohodě

Pro správné pochopení smyslu a role virtuálního terminálu NVT je velmi důležité si uvědomit to, co jsme si již naznačili v 66. dílu - že totiž každý virtuální terminál vždy omezuje "individualitu" konkrétních terminálů, a redukuje jejich vlastnosti a schopnosti na takovou úroveň, která může být společná prakticky všem fyzicky existujícím terminálům. Nejinak je tomu i v případě virtuálního terminálu NVT, který si proto můžeme představit jako největší společný jmenovatel všech ještě použitelných terminálů. Protokol TELNET jej však chápe jen jako "povinné minimum" a připouští, aby se obě strany mohly v konkrétním případě dohodnout "na lepším" - tedy na tom, že mají a jsou schopny používat nějaká rozšíření vůči tomu, co požaduje NVT.

Přitom ale platí zásada, že použití rozšíření (anglicky: **options**) si nelze vynucovat - každá strana má právo vznášet návrhy na jejich použití, ale druhá strana má vždy právo je odmítnout. Díky tomu je možné vystačit i s "hloupými" terminály, jejichž skutečné schopnosti nepřesahují minimum, požadované virtuálním terminálem NVT, a na druhé straně je možné efektivně využít možnosti a schopnosti lépe vybavených terminálů.

Bezprostředně po navázání spojení tedy obě strany mohou používat právě a pouze to, co jim zaručuje virtuální terminál NVT. Mohou však kdykoli zahájit "licitaci", v rámci které se dohodnou na použití oboustranně přijatelných rozšíření. Vzájemné domlouvání na použití různých rozšíření obvykle probíhá okamžitě po navázání spojení, ale není to nutnou podmínkou. Stejně tak se mohou obě strany kdykoli dohodnout na tom, že přestanou určité rozšíření používat. Zajímavá je v tomto ohledu také zásada rovnoprávnosti - každá ze stran má stejné právo navrhnout používání určitého rozšíření, a stejně tak může požadovat i ukončení používání určitého rozšíření (žádosti tohoto typu přitom nesmí být druhou stranou odmítnuty).

Protokol TELNET samozřejmě musí definovat konkrétní způsob, jakým mají obě strany postupovat při vzájemné "licitaci" (anglicky: options negotiation). Příslušný mechanismus je ovšem koncipován jako otevřený - nemůže totiž anticipovat všechna budoucí rozšíření, která budou připadat v úvahu, a tak pro ně nemůže přesně předepisovat konkrétní formu "licitace". Místo toho ponechává otevřený prostor pro individuální postupy vzájemného dohadování, ale současně s tím je umožňuje jednoznačně detekovat, tak aby druhá strana měla vždy možnost rozpoznat, že jde o nabídku používání rozšíření, a i když jí nerozumí, mohla ji odmítnout.

Sedmibitové znaky v osmibitových bytech

Pro kódování jednotlivých znaků používá protokol TELNET znakový kód ASCII. Požaduje, aby tiskárna (resp. zobrazovací zařízení) virtuálního terminálu NVT byla schopna znázornit všech 95 alfanumerických znaků ASCII (s kódy 32 až 127), a z 33 řídících znaků povinně vyžaduje interpretaci znaků NULL, CR a LF. Kromě toho stanovuje přesný význam i pro řídící znaky BEL (Bell), BS (Back Space), HT (Horizontal Tab), VT (Vertical Tab) a FF (Form Feed), a to v souladu s jejich významem v kódu ASCII - ovšem s tím, že interpretace těchto znaků není povinná (tj. tiskárna je nemusí interpretovat vůbec, ale pokud ano, pak jen stanoveným způsobem). Pro ostatní řídící znaky kódu ASCII je stanoveno, že nebudou mít na tiskárnu žádný efekt.

Po klávesnici virtuálního terminálu NVT je naopak požadováno, aby byla schopna generovat všech 128 znaků kódu ASCII (i když některé nemají na tiskárnu žádný efekt). Kromě toho je požadováno, aby klávesnice NVT generovala ještě i několik dalších znaků, které mají význam řídících příkazů protokolu TELNET (o nich bude řeč příště).

Jednotlivé znaky sedmibitového kódu ASCII jsou ovšem přenášeny zásadně v osmi bitech. Díky tomu je pak možné k nim "přidat" ještě i právě naznačené řídící příkazy (odlišené nastavením nejvyššího bitu). Jako jedno z možných rozšíření se ale obě strany mohou dohodnout na tom, že si budou předávat osmibitové znaky (tj. znaky, kódované v osmi bitech). Pak je ovšem nutné zajistit potřebnou transparenci - umožňující jednoznačně odlišit příkazy od "užitečných dat" - jiným způsobem.

ŠIFROVÁNÍ

Enigma

Samostatný oddíl jsem se rozhodl věnovat šifrovacímu stroji **Enigma**, jelikož se jedná o jeden z nejznámějších šifrovacích strojů. Enigmu používali za druhé světové války **německá** vojska k šifrování tajných informací. Její název pochází z latiny a znamená hádanka či záhada.

Enigmu si nechal patentovat **18.2.1918** německý inženýr **Arthur Scherbius** a v dubnu téhož roku ji nabídl německému námořnictvu. V letech **1926** a **1928** německé námořnictvo používalo upravenou verzi stroje Enigma. V roce **1928** se jeden exemplář Enigmy dostal díky problémům při transportu do rukou polským matematikům ve Varšavě. Ti přišli na to, že klávesnice je spojována s kódovacím zařízením v abecedním pořadí. Na základě tohoto zjištění sestrojili dekódovací zařízení **La Bombe**.

Enigma svým vzhledem připomíná klasický **psací stroj**. Před její klávesnicí je umístěno 26 konektorů, které slouží k propojení jednotlivých písmen. Za klávesnicí se pak nachází **svítící deska**, která obsahuje 26 písmen. Pomocí žárovek umístěných na této desce je možno rozsvítit jakékoliv písmeno. Vlastní šifrovací mechanismus se skládá z prostoru, na jehož stranách jsou **dvě kola**, mezi která se vkládají další **tři kola**. Obě krajní kola mají 26 kontaktů, které odpovídají jednotlivým písmenům abecedy. Kolo v levé části se nazývá **reverzní kolo**. Tři

vnitřní kola se vybírají z pěti možných, verze s výběrem z osmi různých kol se používala pouze v námořnictvu. Každé z vnitřních kol obsahovalo **vpravo 26 pružinových** a **vlevo stejný počet plochých konektorů** tak, aby do sebe zapadaly.

Vždy, když je stisknuto písmeno na klávesnici, tak se první kolo otočí o jednu pozici. Poté, co se první okolo otočí 26x, otočí se druhé a nakonec třetí. Dostáváme tak celkem 26x26x26, tj. **17576 různých stavů**. Pro ztížení práce kryptoanalytikům byla délka zprávy omezena na 250 znaků, aby se nemohly opakovat sekvence, což by útočníkovi velice pomohlo při luštění kódu.

Rozdíly mezi symetrickou a asymetrickou šifrou

Ačkoliv na intuitivní úrovni byl rozdíl mezi symetrickým a asymetrickým šifrováním objasněn již v úvodu této práce, rozhodl jsem se pro detailnější popis obou systémů. Pro pochopení práce algoritmů, které jsou popsány dále je nutné poznat výhody i nevýhody těchto dvou kryptografických algoritmů.

Začněme s popisem **symetrického šifrovacího systému**. Svůj název získal dle šifrovacího klíče. Pro šifrování i dešifrování se totiž používá **stejný šifrovací klíč**. Logicky je tedy klíč tedy nutné často obměňovat. Tento problém se řeší generováním klíčů pro sezení, či lépe z angličtiny tzv. **session key**. Odesilatel zprávy vygeneruje šifrovací klíč, pomocí kterého zašifruje otevřený text. Šifrovanou verzi poté veřejným kanálem pošle příjemci, který ji dešifruje pomocí stejného šifrovacího klíče. Nutně zde však vyvstává další otázka: jak poslat šifrovací klíč **tajně** a **bezpečně** příjemci? To je hlavní problém symetrického šifrovacího sytému. Pokud by se třetí straně podařilo získat šifrovací klíč, který je posílán pomocí tajného kanálu, dojde k úniku informací. Výhodou symetrických šifrovacích algoritmů je jejich rychlost. Mezi jednoduché symetrické šifrovací systémy patří například i **Caesarova šifra**.

Asymetrický šifrovací systém si dává za úkol zajistit bezpečnou komunikaci pomocí neutajeného komunikačního kanálu a je též nazýván systémem s **veřejným klíčem**. Ústředním pojmem je **jednocestná funkce**, kterou definoval **R.Needham**: jedná se o funkci **f: x → y=f(x)**, u níž je jednoduché pro všechna **x** vypočítat **y**, ale pro všechna **y** je spočetně nemožné získat **x**. Na základě toho je generována dvojice klíčů: **veřejný** a **soukromý**. Veřejný klíč určité osoby může vlastnit kdokoliv. Pomocí něho můžeme pouze zašifrovat zprávu určenou určitému subjektu. Naproti tomu soukromý klíč si vlastník musí chránit před zneužitím. Pomocí něho je možno dešifrovat přijaté zprávy. Tím je zajištěno, že pouze vlastník soukromého klíče, jemuž je šifrovaná zpráva určena, je oprávněn přijatou zprávu dešifrovat. Z hlediska bezpečnosti je teoreticky možné odvodit z veřejného klíče klíč privátní, ale při dostatečné délce klíče je to v současné době výpočetně nemožné; nicméně uvidíme za několik let…

SYMETRICKÉ ŠIFROVACÍ SYSTÉMY

DES (Data Encryption Standard)

DES je prvním z veřejných kryptografických algoritmů. I přes řadu v současné době známých chyb je velice populární a často používán.

V roce **1973** vyhlásilo ministerstvo obchodu USA soutěž na vytvoření šifrovacího standardu, který by dostatečně zabezpečil ochranu důvěrných dat v informačních systémech. Vysokým nárokům však nevyhověl žádný z navrhovaných algoritmů, a proto se soutěž konala v roce **1974** znovu. Vítězem se stala firma **IBM**. Ta nevyvinula zcela nový algoritmus, ale „pouze“ zdokonalila svůj stávající šifrovací algoritmus **Lucifer**, jenž byl vyvinut výzkumným týmem pod vedením doktora **Tuchmana**. Šifrovací algoritmus DES prošel i bezpečnostním hodnocením **National Security Agency** (též známé pod zkratkou ***NSA***). V roce **1975** si jej firma IBM nechala patentovat. Zároveň umožnila její bezplatné používání na území USA. V březnu téhož roku byl zveřejněn šifrovací algoritmus DES. V listopadu **1976** byl DES přijat jako šifrovací standard pro

zabezpečení neutajovaných dat v civilním a vládním sektoru s předpokládanou délkou používání deset až patnáct let, s podmínkou, že jeho bezpečnost bude každých pět let kontrolována. Díky masovému rozšíření šifrovacího standardu DES však vznikl nový problém: jak ve velkém měřítku nahradit tento šifrovací standard novějším a lepším? DES se totiž stal neoficiálním mezinárodním standardem ve veřejném i soukromém sektoru. Již v roce **1975** se začalo spekulovat o bezpečnosti algoritmu DES. K prvním kritikům patřili **Diffie** a **Hellman**. Ti kritizovali zejména nedostatečnou délku šifrovacího klíče. Argumentovali tím, že k rekonstrukci zašifrovaného textu stačí pouze vyzkoušet všechny možné kombinace šifrovacích klíčů. Na základě jejich výhrad byla svolána konference, která dospěla k závěru, že tento druh útoku v reálném čase bude možné praktikovat až technologiemi vyvinutými po roce 1990.

V roce **1997** vypsala agentura **RSA** kryptoanalytickou soutež, v níž bylo cílem rozluštit text se známým začátkem a délkou šifrovacího klíče 56 bitů. Po necelých pěti měsících byla šifra prolomena. Tímto činem si **Rocke Versen**, vedoucí týmu **DES Challenge**, vydělal 10 000 dolarů a zároveň dokázal reálnou prolomitelnost šifrovacího standardu DES. K dešifrování zprávy využil Internet – s pomocí týmu luštitelů zkoušeli kombinace klíčů, dokud nebyla zpráva čitelná.

Algoritmus šifruje 64 bitů otevřeného textu na 64 bitů šifry. Každý osmý bit je však kontrolní, takže efektivní délka klíče je pouze 56 bitů. Kvůli této relativně nízké délce klíče byl přijat standard známý pod názvem **Triple DES** (TDES, 3DES). Ten se od klasického DES liší tím, že stejná data projdou algoritmem třikrát, čímž se zvýší efektivní délka klíče.

Skipjack

Novým šifrovacím algoritmem Skipjack a systémem smluvního uložení sdílených klíčů vyřešila NSA problém bezpečnosti komerčně používaných šifer v souladu s možností jejich kryptoanalýzy v NSA pro ochranu státních zájmů USA. Principem je implementace tajného klíče do každého čipu typu **Clipper** při výrobě. Tento klíč je rozdělen na dvě poloviny a každá z nich je bezpečně uložena v organizaci, kterou určí stát. Poté je možno na základě soudního příkazu tento klíč vydat a podezřelé informace dešifrovat. Díky tomuto principu NSA v případě potřeby nemusí šifru luštit, ale stačí jí pouze legální cestou získat klíč. V roce **1994** byl schválen standard **EES** (Escrewed Encryption Standard), díky kterému lze čipy Clipper volně exportovat. Kontrola bezpečnosti je zajištěna tak, že se tyto čipy mohou vyrábět pouze ve vládou schválených a kontrolovaných organizacích. Podle posudku NSA je jediným možným útokem na Skipjack postupné vyzkoušení všech kombinací klíčů, což je však v současné době v reálném čase technicky nemožné. Díky principu, na kterém je systém sdílených klíčů založen, vzniká mnoho debat o demokratičnosti tohoto způsobu utajování informací.

CAST

Název tohoto šifrovacího algoritmu vznikl na základě jmen jeho tvůrců, kterými byli **C.Adams** a **S.Tavares**. Důvodem pro vznik tohoto šifrovacího algoritmu byla situace na poli kryptografických algoritmů v devadesátých letech. Šifrovací standard DES již nepůsobil příliš důvěryhodně a kvalitní šifry byly finančně velice náročné. CAST se v současné době velice často používá a stal se „neoficiálním standardem“.

IDEA

Tento algoritmus byl vyvinut ve Švýcarsku jako alternativa k šifrovacímu standardu DES. Jeho název je zkratkou z ***International Data Encryption Algorithm***. Algoritmus byl publikován v roce **1991**, původně však nesl název **IPES**. Jeho autory byli **J.Messey** a **X.Lai**. IDEA je vylepšenou verzí předchozího algoritmu **PES**, u kterého byla publikována metoda prolomení.

ASYMETRICKÉ ŠIFROVACÍ SYSTÉMY

RSA

Algoritmus RSA, jehož pojmenování vzniklo z prvních písmen příjmení jeho autorů (Rivest, Shamir, Adleman), vznikl v roce 1977. Podstatou algoritmu je skutečnost, že je velmi obtížné faktorizovat (rozložit) velká čísla, kort když jsou součinem dvou velkých prvočísel. Přesto bezpečnost RSA je přímo úměrně závislá délce klíče.

Stručná historie RSA

RSA vznikl v roce 1977 na Massachussets Institute of Technology (MIT). V roce 1983 byl na zmíněný algoritmus vydán U.S. Patent (číslo #4,405,829). Nositelem tohoto patentu byla firma RSA Security, kterou autoři RSA mezitím založili. Patent, jehož platnost byla pouze na území USA, byl udělen na 17 let (do roku 2000) a znamenal, že za jakékoliv komerční využití RSA bylo nutné platit firmě RSA Security licenční poplatky.

Algoritmus RSA patří do skupiny tzv. „asymetrických šifer“, kde existuje „veřejný“ a „privátní šifrovací klíč“. Veřejný klíč slouží k šifrování zpráv směrem k uživateli, privátní k jejich dešifrování. Primárně se tento algoritmus prezentuje jako algoritmus pro výměnu klíčů a tvorbu elektronického podpisu.

Algoritmus RSA

Celý algoritmus je tedy založen na obtížnosti faktorizace velkých čísel. Oba klíče se odvozují jako součin dvou velkých (100-200 místných) prvočísel.

$$n = p.q$$

Poté se zvolí šifrovací klíč e tak, aby čísla e a $(p-1).(q-1)$ byla čísla nesoudělná. A pomocí Eulerova rozšířeného algoritmu vypočteme dešifrovací klíč d, pro který platí.

$$e.d = 1(mod(p-1)(q-1))$$

V tuto chvíli již čísla p a q pro další postup nepotřebujeme. Přesto je nikdy nesmíme prozradit, neboť tím bychom oslabili bezpečnost algoritmu.

V tuto chvíli musíme rozdělit zprávu na bloky, které budou kratší než-li n (pokud p a q jsou 100 místná čísla, n bude 200 místné, měly by části zprávy být kratší než-li 200.

A teď můžeme za pomoci tohoto algoritmu vesel šifrovat a dešifrovat.

$$\text{Šifrování: } c = m^e.mod\ n$$
$$\text{Dešifrování: } m = c^d.mod\ n$$

Tím máme celý algoritmus popsán. Paradoxem asymetrických algoritmů obecně je jejich poměrně vyšší rychlost zpracování při softwarové realizaci, než-li při hardwarové. Sice byla vyvinuta řada mikročipů, které realizují RSA, ale jsou při zpracování 1000x pomalejší než-li algoritmus DES. Při softwarové realizaci je algoritmus pouze 100x pomalejší než-li DES.

Luštění algoritmu

Existuje několik velmi složitých způsobů luštění algoritmu RSA. Dá se říci, že obecně je možné uvést několik zásad pro luštění algoritmu: *Znalost jednoho šifrovacího/dešifrovacího páru exponentů daného modulu umožňuje luštiteli faktorizovat tento modul. *Znalost jednoho šifrovacího/dešifrovacího páru exponentů daného modulu umožňuje luštiteli odvodit jiný šifrovací/dešifrovací pár bez nutnosti faktorizace. *Doplnění zpráv náhodnými hodnotami slouží jako prevence proti vyluštění RSA metodou luštění s nízkými šifrovacími exponenty. *Dešifrovací exponent vymel být velký.

Digitální podpis

Algoritmus RSA lze snadnou využít pro digitální podepisování. Princip jeho použití je pak opačný, než-li při šifrování. Účastník přidá ke zprávě identifikační číslo, které vytvořil „dešifrováním“ hashe své zprávy pomocí svého soukromého klíče. Pro ověření pak stačí pouze znovu zašifrovat daný identifikátor pomocí veřejného klíče a výsledky porovnat. Pokud vyjde stejná hodnota, dokument byl opravdu podepsán dotyčným.

RSA v současnosti představuje celosvětovou normu, kterou formuluje i ISO 9796. Například francouzské a australské bankovnictví je normalizováno na bázi RSA. V USA, díky tlaku různých organizací (v neposlední řadě také NSA) neexistuje žádná norma pro šifrování na bázi veřejného klíče. Mimoto, návrh bankovní normy ANSI dosti zpřesňuje použití RSA.

IP Multicast

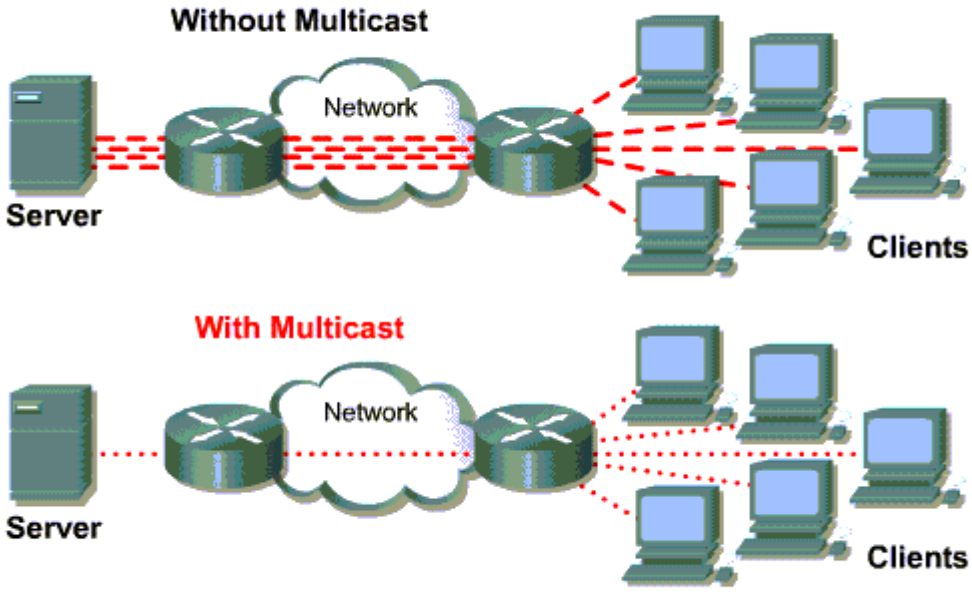
Cíl kapitoly

Cílem kapitoly je pochopit smysl skupinového vysílání (multicastingu) a naučit se jej využívat v prostředí lokální i rozlehlé sítě. Student bude rozumět principům směrování multicastů a dokáže zvolit vhodný směrovací protokol s ohledem na rozložení potenciálních přijímačů jednotlivých multicastových skupin.

Prekrekvizity

Studium této kapitoly předpokládá obecnou znalost problematiky směrování protokolu IP a mechanismu enkapsulace IP paketů v rámcích spojové vrstvy, zejména na Ethernetu.

Princip a aplikace multicastingu



- Výhodou omezení zabrané šířky pásma ve větvích vedoucích k více přijímačům oproti distribuci pro každý přijímač zvlášť.
- Pakety se kopírují až v místech, kde se datový tok rozděluje do více větví sítě obsahujících příjemce multicastové skupiny.

Typické aplikace

- distribuce vysílání videa, hlasu
- videokonference

- vyhledávání služeb
- distribuované simulace (včetně her)

Multicastové skupiny

- Skupiny jsou "otevřené" - může do nich vysílat i stanice, která není členem skupiny
- Do skupiny se může přihlásit kterákoli stanice
- Stanice může být členem i více skupin

Adresování multicastových skupin v IP

Adresy třídy D: 224.0.0.0 - 239.255.255.255

Rezervované lokální adresy

- Rozsah adres 224.0.0.0 - 224.0.0.255
- Šíření multicastů omezeno na lokální segment, pakety vždy TTL=1
- Používáno hlavně směrovacími protokoly (komunikace sousedních směrovačů)

Link Local IP Address	Usage
224.0.0.1	All systems on this subnet
224.0.0.2	All routers on this subnet
224.0.0.5	OSPF routers
224.0.0.6	OSPF designated routers
224.0.0.9	RIP Version 2 routers
224.0.0.10	EIGRP routers
224.0.0.12	DHCP server/relay agent
224.0.0.13	All P1M routers
224.0.0.22	IGMP
224.0.0.25	Router-to-switch (such as RGMP)

Adresy s globálním dosahem (Globally Scoped Addresses)

- rozsah 224.0.1.0 - 238.255.255.255
- přiděluje IANA (např. 224.0.1.1 - Network Time Protocol (NTP))

GLOP adresy

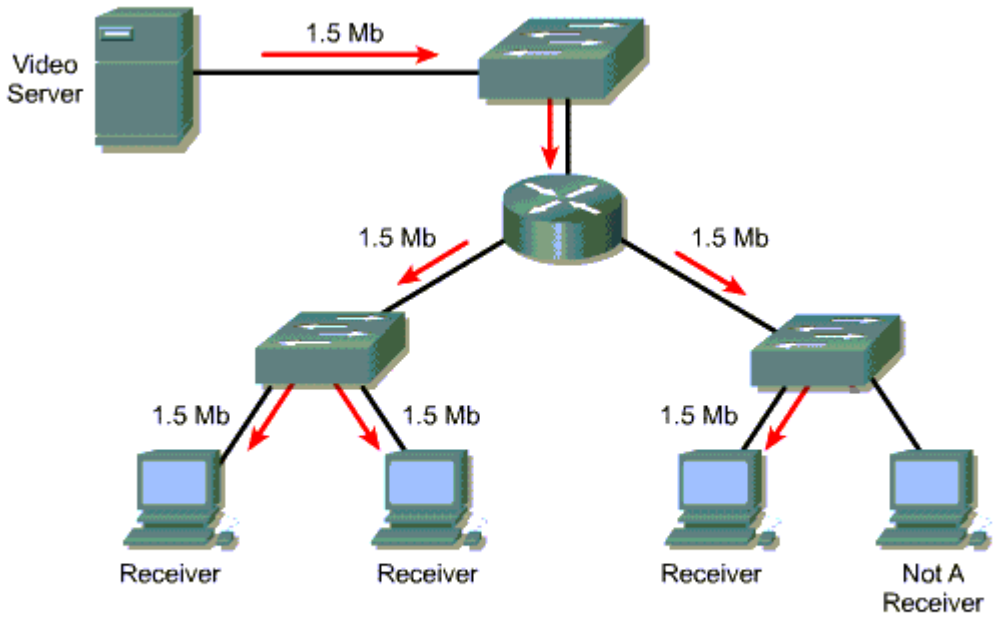
- rozsah 233.0.0.0 - 233.255.255.255
- automaticky přiděleny sítím, které mají své číslo AS
- číslo AS zakódováno ve 2. a 3. bajtu
- poslední bajt umožňuje v AS zřídit až 255 lokálních multicast skupin

Adresy s omezeným dosahem (Limited Scope Addresses, RFC 2365)

- rozsah 239.0.0.0 - 239.255.255.255
- obdoba privátních IP adres, mohou být opakovaně využívány v nezávislých sítích
- oblast sítě, kam se šíří, vymezená konfigurací směrovačů

Poznámka
Multicastové adresy mohou být v IP paketech jen adresami **cíle** (zdrojová adresa je vždy unicast)

Multicasting na LAN a WAN



Problémy:

- Mapování IP adres multicastových skupin na MAC adresy (neřeší ARP)
- Šíření multicastů (kopírování paketů) jen do větví sítě, kde je zájem o příjem příslušné skupiny

Šíření ve stromové sítí (L2):

Kopie na porty (mimo příchozího) vycházející do větví, ve kterých existuje nějaký příjemce dané skupiny.

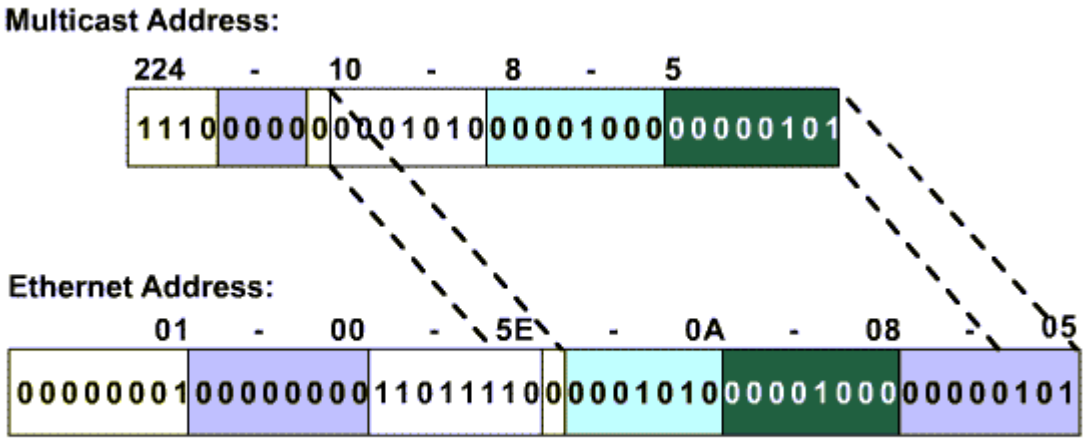
Šíření v síti obsahující smyčky:

- Smyčky by vedly k cyklování paketů, proto konstrukce distribučního stromu.
- Kořenem distribučního stromu buďto směrovač sítě se zdrojem multicastů nebo dohodnutý směrovač (kořen, RP-Rendezvous Point,Core). Ve druhém případě zdroj zasílá multicastové pakety unicast IP tunelem na kořenový směrovač.
- Směrovače musí rozlišit, která rozhraní vedou ke kořeni stromu a která ke větvím, proto pro multicasty existuje zvláštní forma směrovací tabulky.

Mapování multicast IP adres na MAC adresy (Ethernet)

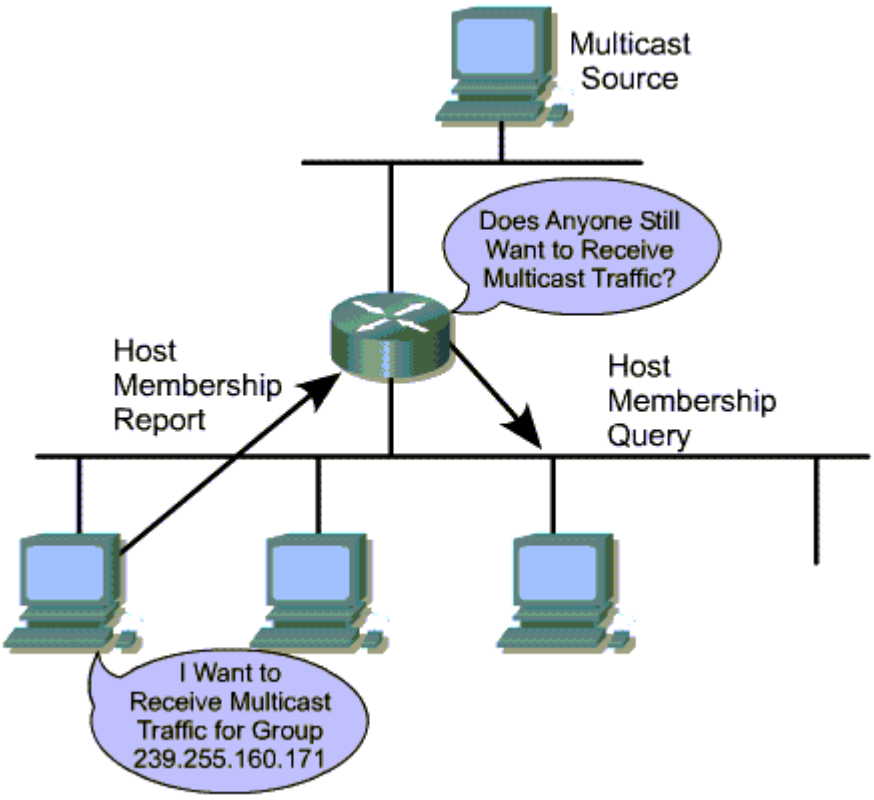
- Pro multicast MAC adresy vyhrazena polovina rozsahu dáného prefixem 0x01-00-5E (všimněte si, že příznak "multicast" v nejvyšším bitu je hodnotou 0x01 nastaven)
- V v nejvyšším bitu 4. bajtu MAC adresy musí být 0, pro mapování zbývá 23 bitů MAC adresy
- Multicast IP adresy (třída D) vždy začínají 1110, musí se mapovat 28 bitů (32-4)
- Posledních 23 bitů multicast IP adresy se přímo zkopíruje do posledních 23 bitů MAC adresy
- 5 nejvyšších bitů multicast IP adresy se nemapuje, takže 32 IP multicast skupin se vždy mapuje na jednu multicast MAC adresu (mapování není jednoznačné)

Příklad: 224.10.8.5 -> 01.00.5e.0a.08.05



Internet Group Membership Protocol (IGMP)

IGMPv1 - RFC 1112, IGMPv2 - RFC 2236



- používají stanice (přijímače multicastu) pro přihlášení (u v2 i odhlášení) do multicastové skupiny u nejbližšího směrovače a směrovač pro ověření, zda ve větvi ještě existuje stanice se zájmem o danou multicastovou skupinu
- používají směrovače pro přihlášení své větve sítě do distribučního multicastového stromu
- směrovače přes IGMP získávají informaci, zda je na jednotivých rozhraních nějaký (aspoň jeden) přijímač jednotlivých multicastových skupin

Protokol IGMP v.1

Protokol IGMP v.1 obsahuje pouze dvě zprávy:

- MEMBERSHIP REPORT - zasílá stanice při přihlášení do skupiny a jako pozitivní odpověď na dotaz směrovače, zda je na segmentu zájemce o danou skupinu. Zasílá se na adresu požadované multicast skupiny.
- MEMBERSHIP QUERY - periodicky zasílá směrovač na každém rozhraní pro zjištění, zda je na segmentu nějaká stanice se zájmem o některou multicastovou skupinu.

Pokud stanice přestane mít zájem o multicast skupinu, přestane na dotaz směrovače odpovídat zprávou MEMBERSHIP REPORT.

Poznámka:
Na segmentu sítě s velkým množstvím přijímačů multicastové skupiny by dotaz směrovače zprávou MEMBERSHIP QUERY vygeneroval smršť odpovědí MEMBERSHIP REPORT. Proto je do protokolu vnesena optimalizace: stanice neodpovídá na zprávu MEMBERSHIP QUERY ihned, ale jistou náhodnou dobu s odpovědí posečká. Pokud uslyší, že mezitím na dotaz odpověděla některá jiná stanice, zprávu u MEMBERSHIP REPORT sama generovat nebude.

Protokol IGMP v.2

Novější verze protokolu IGMP v.2 přináší novou zprávu LEAVE GROUP, kterou se stanice může explicitně odhlásit z multicastové skupiny. Výhodou je zkrácení latence odhlašování, tedy doby, po kterou je segment zbytečně zahlcován provozem multicastové skupiny, kterou již nikdo neposlouchá.

Dalším rozšířením verze 2 je možnost specifického dotazu na zájemce o jednu konkrétní multicast skupinu ve zprávě MEMBERSHIP QUERY (nikoli jen na zájemce o jakoukoli skupinu, jako v IGMP v.1). To je užitečné zejména když router přijme zprávu LEAVE GROUP pro nějakou multicastovou skupinu a chce bezprostředně poté ověřit, zda má ještě stále smysl na segment provoz dané skupiny vysílat.

Poznámka:
Všimněte si, že kvůli možnosti havárie stanic bez odhlášení ze skupiny a také kvůli optimalizaci ve vysílání odpovědí na MEMBERSHIP QUERY pouze jedinou stanicí není možné, aby směrovač počítal počet přijímačů na segmentu pouze na základě zvyšování, resp. snižování čítače při příchodu zpráv MEMBERSHIP REPORT a LEAVE GROUP pro danou skupinu. Proto je nezbytné, aby nepřítomnost přijímačů pro danou skupinu zjišťoval dotazem pomocí zprávy MEMBERSHIP QUERY.

Protokol IGMP v.3

Nejnovější specifikace protokolu IGMP obohacuje protokol o možnost filtrování požadovaného provozu na základě zdrojových adres. Tím se je možné bránit před nekorektně se chovajícími stanicemi, které multicastovou skupinu nežádoucím způsobem zahlcují. Při žádosti o provoz multicastové skupiny je potom možné explicitně zadat buďto zadat seznam zdrojů, ze kterých jedině mají být zprávy přebírány nebo naopak seznam zdrojů, od nichž mají být zprávy zaslané do multicastové skupiny filtrovány.

Zpracování multicastů v aktivních prvcích 2. vrstvy

Jelikož multicast pakety se šíří v rámcích 2. vrstvy, je třeba, aby přepínače věděly, na které porty mají příchozí multicast rámec zaslat. Jednodušší přepínače, které multicast nepodporují, pracují s multicasty stejně jako s broadcasty - zasílají je na všechny porty mimo příchozího. Tím se síť značně zatěžuje, protože všechny multicasty na segmentu sítě (broadcast doméně) jsou zasílány i do těch větví, kde nejsou požadovány žádnou stanicí.

OBRAZEK.

Je proto třeba, aby přepínače věděly, na kterém portu jsou příjemci které multicast skupiny. To se lze nejlépe dozvědět z protokolu IGMP, včetně monitoringu MAC adres stanic, si které zprávy protokolu IGMP vyměňují. Protože známe způsob mapování multicast IP adres a multicast MAC adresy, postačí přepínači zjistit, na které (unicast) MAC adresy stanic má rozkopírovat příchozí rámec s každou jednotlivou multicast MAC adresou. Čísla portů, na které jsou stanice s těmito MAC adresami připojeny, se již najdou v normální přepínací tabulce.

Poznámka:
Všimněte si však, že sledování zpráv protokolu IGMP je mimo rozsah běžných "kompetencí" přepínače.

Existují dva základní způsoby, jak může přepínač informaci z IGMP získat:

- IGMP Snooping
- pomocný protokol mezi přepínačem a směrovačem segmentu

IGMP Snooping

IGMP Snooping spočívá v tom, že přepínač nahlíží do dat 3. vrstvy v přenášených rámcích a pokud je v ní obsažená zpráva IGMP (MEMBERSHIP REPORT, LEAVE GROUP), získá z ní potřebnou informaci o příslušnosti MAC adres stanic do jednotlivých multicastových skupin. Tato metoda samozřejmě přepínač zatěžuje a zpravidla má smysl jen v případě, že že je realizována hardwarově.

Poznámka
Všimněte si, že se zprávy MEMBERSHIP REPORT a LEAVE GROUP podle protokolu IGMP zasílají na adresu příslušné IP multicastové skupiny a tedy na 2. vrstvě na multicastovou MAC adresu, na kterou se daná IP multcastová adresa mapuje. Postačí tedy, aby přepínač zkoumal 3. vrstvu jen u rámců zasílaných na multicastové MAC adresy.

Pomocný protokol mezi směrovačem a přepínači segmentu

Aplikace pomocného protokolu mezi přepínačem a směrovačem segmentu je výhodná tím, že přepínač není zatěžován nahlížením do 3. vrstvy a může být tak aplikována i na méně výkonných přepínačích. O informování všech přepínačů o MAC adresách stanic aktuálně přihlášených do jednotlivých multicastových skupin se stará směrovač příslušného segmentu, který protokol IGMP zpracovává. Protokol, kterým směrovač přepínače na segmentu o mapování MAC adres stanic do jednotlivých skupin informuje, však bohužel není standardizován. Musí tedy být použit přepínač a směrovač téhož výrobce, pokud však výrobce vůbec tuto metodu podporuje. Například firma Cisco pro tento účel implementuje svůj vlastní protokol CGMP (Cisco Group Management Protocol). V něm je rezervována speciální multicast MAC adresa, na kterou směrovač s aktivovaným protokolem CGMP na příslušném rozhraní informuje všechny směrovače segmentu o událostech zjištěných pomocí IGMP na tomto rozhraní.

Distribuční stromy

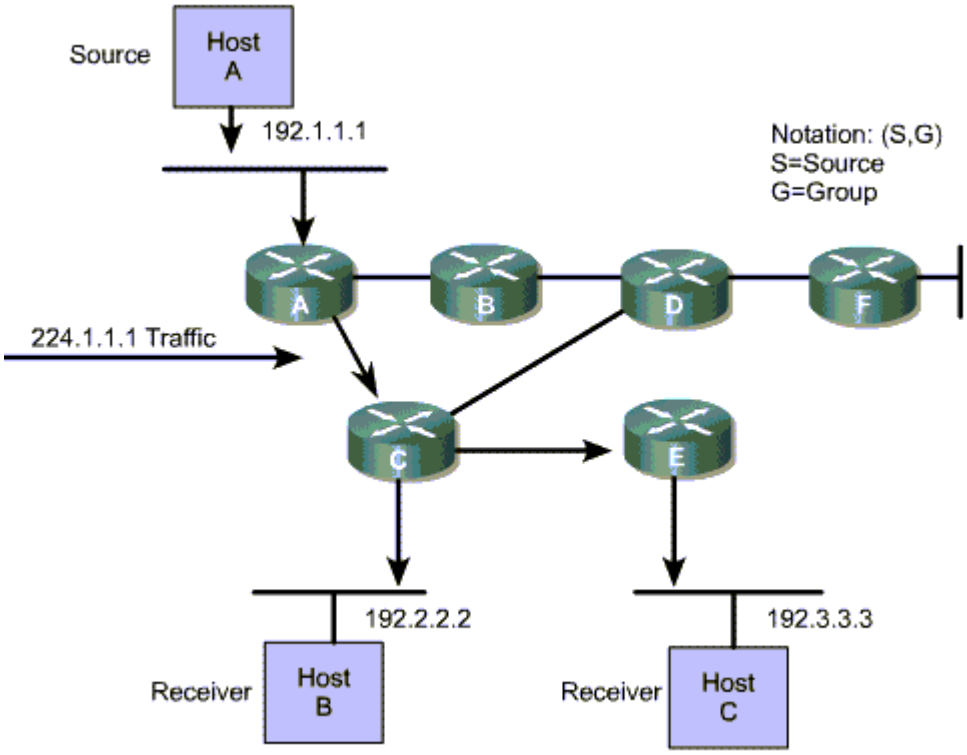
Distribuční strom tvoří acyklický podgraf grafu topologie sítě a spojuje všechny sítě, na nichž jsou přijímače dané multicastové skupiny. Jak se do skupiny přihlašují nové přijímače a odhlašují existující, musí se do stromu připojovat nové větve a odstraňovat ("prořezávat", angl. "prune") staré.

SPT (Shortest-Path Tree, Source Tree) Sdílený strom (Shared Tree) Šíření ve stromu buďto jednosměrné (od kořene k listům) nebo obousměrné

SPT (Shortest-Path Tree, Source Tree)

- Strom nejkratších cest ze zdroje ke všem sítím s přijímači
- Značení < S,G >, kde S je zdrojová adresa, G je adresa multicastové skupiny
- Existuje samostatný strom pro každý zdroj vysílání do každé multicastové skupiny
- => Optimální, ale málo škálovatelné řešení

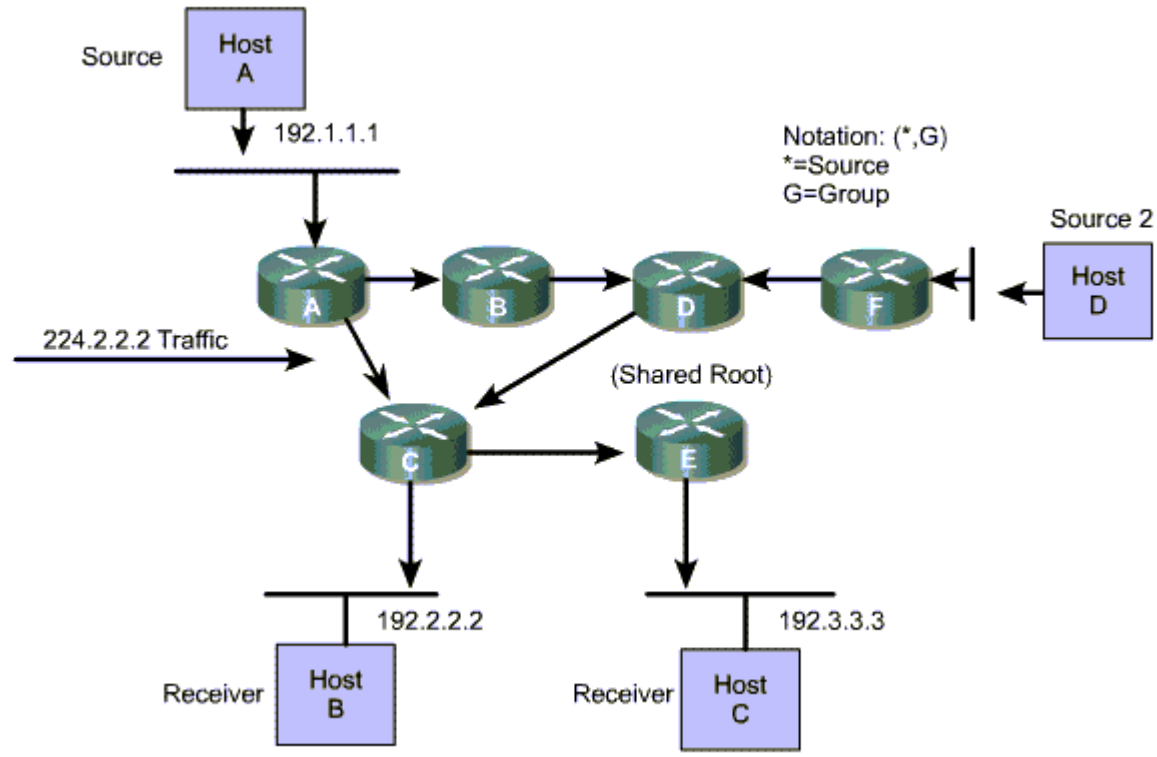
Příklad: strom < 192.2.2.2, 224.1.1.1 >



Sdílený strom (Shared Tree)

- Strom společný pro všechny zdroje ve skupině
- Kořenem je vhodně zvolený směrovač (RP)
- Zdroj musí zasílat každý paket na kořen
- Značení < *,G >, kde * označuje všechny zdrojové adresy vysílající do skupiny G
- => Poněkud delší cesta paketů (navíc cesta ke kořeni), ale jen jediný strom pro skupinu

Příklad: strom < *, 224.2.2.2 >



Směrování multicastů

Na rozdíl od běžného směrování se směrování multicastů děje podle **zdrojové adresy** . Cílem je šířit paket po distribučním stromu stále dál od zdroje vysílání.

Reverse Path Forwarding

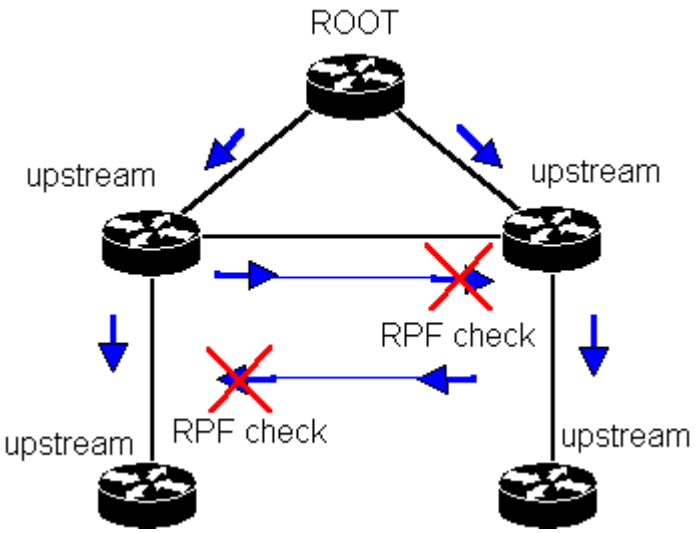
Reverse Path Forwarding je metoda šíření multicast paketů od zdroje vysílání dolů po distribučním stromu. K určení rozhraní, kterými se má příchozí multicast paket dále šířit, slouží běžná (unicast) směrovací tabulka.

Z pohledu distribučního stromu pro danou multicastovou skupinu (a případně pro daný zdroj u < S,G &gtstromů) dělíme rozhraní každého směrovače na "downstream" a "upstream". Upstream rozhraní je právě jedno a vede směrem ke kořeni distribučního stromu. Je to rozhraní, kterým se vysílají běžné (unicast) pakety na adresu odpovídající kořeni distribučního stromu. Downstream rozhraní jsou rozhraní, která vedou do segmentů sítě v nichž se vyskytují příjemci dané multicastové skupiny, ať už přímo stanice nebo směrovače, které mají takovté příjemce za svými downstream rozhraními.

Aby bylo zajištěno šíření každého multicast paketu od kořeni k listům distribučního stromu, nezpůsobili cyklování multicast paketů ve smyčce a pokud možno vyloučili vícenásobné vysílání téhož multicast paketu na segment, postupuje směrovač při přijeté multicast paketu podle následujího pravidla (tzv. RPF Check):

Pokud paket přišel z rozhraní, které se podle směrovací tabulky používá pro směrování paketů ke zdrojové adrese multicastového paketu, paket se rozešle na downstream rozhraní. V opačném případě se paket zahodí.

Pokud na základě RPF check směrujeme paket na všechna rozhraní mimo příchozího, vyloučí se cyklování paketů ve smyčce a zajistí jeho rozšíření do celé sítě. Nezajistí se tim však, že paket neprojde některou síti vícekrát, jak je uvedeno na obr. X.



Pro vyloučení tohoto případu multicastové směrovací protokoly určují ke každému distribučnímu stromu nejen upstream, ale i množinu downstream rozhraní. Pakety vyhovující podmínce RPF check se pak neposílají na všechna rozhraní mimo příchozího, ale pouze na downstream rozhraní. Přiřazení upstream a downstream rozhraní každému distribučnímu stromu tak tvoří směrovací tabulku pro multicasty, jejíž záznamy mají pro zdrojové stromy (source trees) tvar:

< cílová multicast adresa, adresa zdroje multicastu, upstream rozhraní, množina downstream rozhraní >

Záznamů musí být tolik, kolik je vysílačů v multicast skupině.

Pro sdílené stromy (shared trees) se uchovává pro celou skupinu pouze jediný záznam. Upstream rozhraní je určeno rozhraním, z něhož vychází nejkratší cesta k RP. Záznam směrovací tabulky pak má tvar

< cílová multicast adresa, upstream rozhraní, množina downstream rozhraní >

Některé multicastové směrovací protokoly používají nejprve sdílený strom a až v případě příliš intenzivního provozu z některé zdrojové adresy (S) zřídí pro danou multicastovou skupinu a tento zdroj < S,G > strom. Pokud tedy existuje v multicastové směrovací tabulce záznam < S,G > i < *,G > a multicastový paket vyhoví oběma z nich, bude se směrování provádět podle < S,G > záznamu.

Směrovací protokoly pro multicast

Podle režimu funkce dělíme směrovací protokoly do dvou základních skupin:

- Dense Mode - protokoly DVMRP, PIM-DM, MOSPF
- Sparse Mode - protokoly PIM-SM, CBT

Rozdíl spočívá v tom, jestli je předpokládáno hustý (dense) nebo řídký (sparse) výskyt příjemců multicast provozu v jednotlivých větvích sítě. Dále si vysvětlíme základní princip práce v hustém a řídkém režimu.

Hustý režim

V hustém režimu se předpokládá, že zájemci o multicastovou skupinu jsou skoro na všech segmentech sítě. Proto je provoz implicitně směrován do všech segmentů. Pokud směrovač časem (na základě protokolu IGMP) zjistí, že na žádném svém rozhraní nemá zájemce o multicastovou skupinu, pošle směrovači ve vyšší úrovni distribučního stromu žádost o "prořezání" (prune). Směrovač ve vyšší úrovni stromu na jejím základě přestane do větve multicastový provoz dané skupiny vysílat. Pokud se ve větvi časem objeví zájemce o multicastovou skupinu, může lokální směrovač segmentu větev do distribučního stromu opětovně přihlásit.

Řídký režim

Na rozdíl od hustého režimu nejsou v řídkém režimu implicitně multicast pakety zasílány nikam. Většina větví by totiž z důvodu řídkého výskytu příjemců byla multicasty zaplavována zbytečně. Až když se na segmentu objeví zájemce o danou skupinu, pošle se směrem ke kořeni žádost o připojení nové větve distribučního stromu. Po této větvi pak k novému zájemci poteče provoz požadované multicastové skupiny.

Připojení větve do distribučního stromu v řídkém režimu nebo její odpojení v režimu hustém není trvalé. Každý směrovač distribučního stromu si totiž žádost o připojení, resp. odpojení pamatuje pouze po jistou omezenou dobu a po jejím vypršení přejde k původnímu chování. Proto se musí žádost o připojení a odpojení větve distribučního stromu periodicky opakovat.

Žádosti o připojení a odpojení větve stromu se mezi směrovači zpravidla zasílají prostřednictvím zpráv protokolu IGMP. Z hlediska směrovače tak není rozdílu, jestli mulicastový provoz na rozhraní vyžaduje na segmentu připojená stanice nebo jiný směrovač, který multicastový provoz požaduje pro segmenty na jeho rozhraní připojené větve distribučního stromu.

V řídkém režimu se na řízení směrování účastní méně směrovačů-pouze ty, které vytvářejí ne příliš rozvětvený distribuční strom. Proto jsou protokoly pracující v řídkém režimu vhodnější pro rozlehlé WAN sítě.

Směrování řídících zpráv

V hustém i řídkém režimu směrování se musí šířit řídící zprávy, obsahující žádosti o připojení větve do distribučního stromu nebo naopak o její odpojení. Z pohledu distribučního stromu se tyto zprávy šíří od listů směrem ke kořeni. Je tedy třeba zajistit směrování těchto zpráv na všech směrovačích distribučního stromu.

Protože distribuční strom je vždy stromem nejkratších cest ze zdroje nebo z RP k listovým směrovačům, lze pro směrování řídících zpráv použít běžné směrovací tabulky. Řídící zprávu tak zašleme o úroveň distribučního stromu výše tím, že zvolíme rozhraní, kterým bychom se podle směrovací tabulky dostali nejkratší cestou ke kořeni distribučního stromu.

Směrovací protokoly pro pro hustý režim

Distance Vector Multicast Routing Protocol (DVMRP) - RFC 1075

- první reálně používaný protokol pro směrování multicastů
- multicast pakety kopíruje na všechny interface mimo toho, které vede ke zdroji multicastu (reverse-path flooding)
- pruning na základě žádosti z větví bez přijímačů multicastové skupiny
- periodicky se obnovuje flooding do prořezaných větví pro případ, že by se ve větvi objevili přijemci
- pro určení upstream interface DVMRP implementuje svůj vlastní unicast distance-vector směrovací protokol podobný RIPu založený na počtu přeskoků. Maximální počet přeskoků 32, periodický update každých 60 s.

Protože multicast je směrován na základě svého vlastního směrovacího potokolu, může multicastový provoz mezi sítěmi procházet po jiných cestách než normální unicast.

Protocol independent multicast dense mode (PIM-DM)

- nezávislý na používaném unicast směrovacím protokolu, využívá jeho směrovací tabulku
- multicasty směruje na základě RPF check
- není ve skutečnosti směrovacím protokolem, neposílá a nepřijímá žádné routing updates
- flooduje provoz do všech větví; větev může požádat o prořezání, pokud neobsahuje příjemce multicast skupiny (refresh každé 3 minuty)

Podmínky pro efektivní funkci:

- přijímače a vysílače blízko
- málo vysílačů a mnoho příjemců
- intenzivní multicastový tok
- konstantní proud multicast dat

Multicast OSPF (RFC 1584)

- směrování multicastů závislé na použitém směrovacím protokolu (OSPF)
- použitelné v rámci jedné směrovací domény (OSPF)
- šíření informací o členství ve skupinách na jednotlivých segmentech v link state updates.
- pro prostředí s nevelkým počtem současně aktivních skupin a zdrojů multicastů (< S,G > párů)
- každý směrovač provádí nezávislý výpočet distribučního stromu pro každý pár < S,G >

Směrovací protokoly pro pro řídký režim

Core-Based Tree (CBT) - RFC 2201

- jediný sdílený distribuční strom pro skupinu
- kořenem stromu je "core" směrovač, ostatní směrovače posílají žádosti o připojení do stromu směrem k core směrovači
- když core router přijme žádost o připojení větve, pošle zpět acknowledgement, ten ve směrovačích na cestě vytváří stavovou informaci o nové větvi
- pokud žádost o připojení na cestě k core směrovači narazí na směrovač, který je již součástí požadovaného stromu, tento směrovač žádost potvrdí namísto core směrovače
- dosud ve vývoji, tři vzájemně nekompatibilní verze

Protocol independent multicast sparse mode (PIM-SM)

- vhodný pro malý počet příjemců ve skupině a pro nepravidelné datové toky
- odesílatelé zasílají multicast pakety na RP
- příjemci se registrují u RP
- Směrovače na cestě mohou optimalizovat trasu datového toku od zdroje k příjemci

Rendezvous point (RP) pro jednotlivé multicast skupiny musí být nakonfigurován ve všech směrovačích. Existují také (zatím nestandardizované) protokoly, které šíří informaci o aktivních RP pomocí multicastingu.

Poznámka:
PIM může pro některé skupiny pracovat v dense módu a pro některé ve sparse módu.

Designated router lokální sítě

Pokud je na LAN připojeno více směrovačů, je třeba, aby zprávy IGMP MEMBERSHIP QUERY zasílal jen jeden z nich. Stejně tak je třeba, aby interakci s protokolem PIM vně sítě zajišťoval jen jeden z nich. Směrovačem pověřeným těmito činnostmi (tzv. designated router) bude směrovač s nejvyšší IP adresou. Takovíto sousedé schopní pracovat s protokolem PIM se nalézají pomocí zpráv PIM QUERY vysílaných periodicky (každých 30 s) na adresu 224.0.0.2 (all-routers).

Internet Multicast Backbone (MBONE)

Ne všechny směrovače Internetu podporují multicasting. MBONE je experimentální multicastová páteř Internetu.

Fast retransmission (Rychlé obnovení přenosu)

Při náhlém přerušení přenosu se systém snaží toto přerušení obsloužit na nejbližším možném místě a v nejkratším čase. Pokud je to možné, pokusí se obnovit i přenášená data Opět je princip založen pouze na programovém vybavení.

* Při detekci tří duplicitních ACK se vyšle znovu segment, o němž se předpokládá, že je ztracený, aniž by se čekalo na timeout.
* Předpokladem použití Fast Retransmit je, aby přijímač při příchodu segmentu mimo pořadé gemeroval okamžitě duplicitní ACK.
* Fast Retransmit se samozřejmě použije jen tehdy, když jsou zmíněné tři duplicitní ACK přijaty do normální retransmise příslušného segmentu od vypršení timeoutu.
* Fast Retransimit je užitečný pro velká okna - nemusí se opakovat tak velké množství dat, jako by se opakovalo při postupném vypršení timeoutu všech po ztraceném segmentu vyslaných a kvůli němu nepotvrzených segmentů.

Syndrom hloupého okna (Silly Window Syndrome)

K syndromu hloupého okna dochází, pokud přijímač opakovaně nabízí krátké přijímací okno, namísto aby počkal, až bude moci nabídnout větší. Syndrom vede k neefektivnímu vysílání po krátkých segmentech.