Evropský polytechnický institut, s.r.o. Kunovice

Študijný odbor: Elektronické počítače

POČÍTAČOVÉ SIETE

(Bakalárska práca)

Autor: Miloslav Kátlovský
Vedúci práce: Ing. Roman Filkorn

Bratislava júl 2005

Evropský polytechnický institut, s.r.o.

Osvobození 699, 686 04 Kunovice
☎ a fax: 572549018, 548035, e-mail: epi@vos.cz
http://www.vos.cz/epi

Student(ka)
Miloslav Kátlovský
Benediktiho 1
811 05 Bratislava

VÁŠ DOPIS ZNAČKY / ZE DNE	NAŠE ZNAČKA	ZODP.VEDOUCÍ/VYŘIZUJE	KUNOVICE
	BP_EP04/05	Ing. Dušek/Čápková	24.1.2005

Zadání bakalářské práce

Vážený studente, vážená studentko,

jako téma Vaší bakalářské práce ve studiu oboru Elektronické počítače Vám zadávám

Počítačové sítě

Osnova:1. Analýza požadavků zadavatele na počítačovou síť
2. Návrh síťového řešení – softwarového a hardwarového vybavení
3. Regulace přenosu dat v počítačové síti
4. Realizace – fyzické zapojení, konfigurace systémů
5. Hodnocení uživatelů

Bakalářská práce bude zpracována pro: Společenství vlastníků bytových a nebytových prostorů Benediktiho 1-5

Tento dokument je součástí Vaší bakalářské práce.

S pozdravem

Ing. Oldřich **Kratochvíl**
rektor

Evropský polytechnický institut
s. r. o.
Osvobození 699
686 04 KUNOVICE

Prehlásenie autora

Prehlasujem, že som bakalársku prácu vypracoval samostatne na základe vlastných vedomostí a zistení. Použitú literatúru a iné pramene, s ktorými som pracoval alebo citoval, uvádzam v zozname použitej literatúry a zdrojov informácií.

Bratislava júl 2005

5.2 Hardwarové prvky použité v sieti

5.2.1 Zapojenie pasívnych prvkov siete

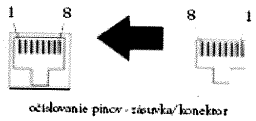
Na vybudovanie pasívnej siete a natiahnutie kabeláží

Bolo potrebné zakúpiť:

- 500m štruktérovanej kabeláže
- 25 ks zásuviek
- 100 ks konektorov a 100 ks kritiek na koncové konektory
- 1 ks 19" rack – skriňu na sieťové komponenty
- 1 ks 48 portový PATCH panet – na prepojenie so zasuvkami v bytoch

Jednotlivé linky štruktúrovanej kabeláže boli pripájané do konektorových koncoviek nasledovným spôsobom:

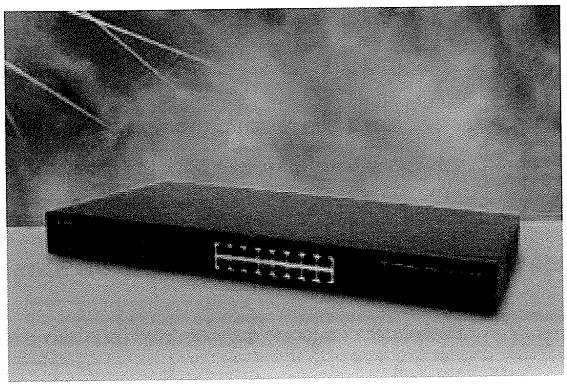
T568A		Prenášaný signál	T568B	
Pin	Farba		Pin	Farba
1	bielo-zelená	TD+	1	bielo-oranžová
2	zelená	TD-	2	oranžová
3	bielo-oranžová	RD+	3	bielo-zelená
4	modrá	-	4	modrá
5	bielo-modrá	-	5	bielo-modrá
6	oranžová	RD-	6	zelená
7	bielo-hnedá	-	7	bielo-hnedá
8	hnedá	-	8	hnedá



očíslovanie pinov - zásuvka/konektor

Obr 7. - Schéma zapojenia štruktúrovanej kabeláže podľa ktorej boli konektorované konsovky

5.2.2 Fast Ethernet přepínač NOVA SWITCH



Obr 8. - 16-portový prepínač PLANET FNSW-1601

Riešenie pre zaťažené siete typu client/server i peer-to-per. Veľmi prijateľné cenové podmienky a jednoduchý prechod medzi 10Mbps a 100Mbps sieťou. Prepínače NOVA Switch kombinujú 16 až 32 portov prepínaného Fast Ethernetu. Rovnako tak sú k dispozícii i modely s jedným až ôsmimi optickými portami Fast Ethernetu. Všetky porty prepínačov sú plne duplexné, Fast Ethernetové porty sú vybavené automatickou detekciou 10/100Mbps.

Technická špecifikácia použitého prepínača FNSW-1601 firmy PLANET

- **16 portov** 100 Base-TX, Switching, Full Duplex (100/200 Mbps), Auto-Negotiation (10/100 Mbps)
- 8k MAC adries, buffer 32kB na každý port
- 4k MAC adries, buffer 32kB na každý port
- 1 uplink port (MDI II) 100Base-TX
- LED diagnostika
- prevedenie rackmount (19") alebo inštalácia stôl (desktop)
- interný zdroj 220/50Hz

5.3 Konfigurácia počítača - router/firewall

Starší stolový počítač:

- Procesor Intel pentium

- 2x sieťový adaptér 100/10Mbit/s – s integrovanou funkciou ITR[9]
- 256 MB RAM
- 20 GB HDD

Táto lacná konfigurácia, je dostatočne výkonná na obslúženie 25 staníc v počítačovej sieti a vykonávať nad nimi ochranný firewall a kontrolu prenášaných dát

5.4 Pripojenie do siete internet cez mikrovlnné spojenie

Tab. 4. - Porovnanie vlastností bezdrátových pripojení v Bratislave

Bluetooth	2 Mb/s	2,4 GHz	slabý výkon, rýchlosť	cena, podpora zariadení, univerzálnosť
802.11	2 Mb/s	2,4 GHz	rýchlosť	stabilná prenosová rýchlosť
802.11a	54 Mb/s	5 GHz	nedostupné zariadenia, cena	nezarušené pásmo, rýchlosť, bezpeznosť
802.11b	11 Mb/s	2,4 GHz	preplnené pásmo, veľmi nízka bezpečnosť	cena, maximálna podpora v OS Linux
802.11b+	22 Mb/s	2,4 GHz	preplnené pásmo	vyššia bezpečnosť
802.11g	54 Mb/s	2,4 GHz	preplnené pásmo	rýchlosť

Prameň: HEČKO, M. *Wi–Fi už nie je len vízia. PC REVUE,* 2003, roč 11, č. 10, str. 24-25

Pripojenie nám sprostredkovala firma SWAN, a.s.

Ako sa nám potvrdilo z praxe, v Bratislave a hlavne v centre mesta je verejné pásmo 2,4GHz preplnené. Po pripojení bez drátového spoju sa nám z plánovaných 2 MBit/s podarilo dostať maximálne na 1,3 MBit/s.

Preto, sme museli prejsť na mikrovlnné spojenie tiež tej istej prenosovej šírky ale na pásme 5,6 MHz, ktoré je navyše bezpečnejšie. Vyjednali sme výhodné cenové podmienky, preto cena nebola výraznou zmenou voči plánu.

[9] Interrupt Throtling

Poďakovanie

Chcem poďakovať kolegom a spolupracovníkom na tomto projekte, predovšetkým svojmu vedúcemu práce Ing. Romanovi Filkornovi a kolegovi Jakubovi Soboňovi za pomoc, odborné rady a pri budovaní siete a inštalácii aktívnych prvkov siete.

Bratislava november 2004

Miloslav Kátlovský

OBSAH

ÚVOD

Zadanie, na vytvorenie počítačovej siete v rámci obytných domov na Benediktiho ulici na stretnutí vlastníkov bytových a nebytových priestorov. Riešili sa možnosti znižovania nákladov na bývanie a služby stým spojene. Kľúčovým bodom schôdze bolo šetrenie nákladov na tepelnú energiu a v rámci návrhov padla aj možnosť ušetriť náklady a internetové pripojenie každého bytu, jedným spoločným pripojením pre niekoľko obytných domov. Ako jeden s obyvateľov týchto domov som sa ujal tohto projektu a riešil ho ako diplomovú prácu.

Mojou úlohou je vybudovať počítačovú sieť a vybrať vhodné internetové pripojenie pre spoločenstvo vlastníkov bytov a nebytových priestorov 1-5. Budem sa zaoberať problematikou výberu, návrhu, a inštalácie fyzických častí siete ako aktívnych tak aj pasívnych prvkov, bezpečnosti počítačovej siete a reguláciou prenosu dát na jednotlivých užívateľov.

V úvode budem analyzovať potreby a požiadavky užívateľov. Cenovú dostupnosť, výhody a prípadné nevýhody spoločného pripojenia na Internet. Rozoberiem možnosti neskoršieho využitia počítačovej siete v rámci bytových domov.

V druhej časti rozviniem problematiku návrhu, bezpečnosti, a regulácie dát počítačovej siete. Budem sa venovať princípom a metódam využívaným v praxi pri komerčných projektoch. Odôvodním výber daného technického riešenia v závislosti od požiadaviek zadávateľa.

V tretej časti tejto práce popíšem konfiguráciu a zapojenie finálneho riešenia, problémy a úskalia pri realizácii.

V nasledujúcej uvediem hodnotenie užívateľov a ich postoje k riešeniu projektu.

1 ANALÝZA PROBLEMATIKY A POŽIADAVIEK

1.1 Popis súčasného stavu a priestorov budov

Spoločenstvo vlastníkov bytov a nebytových priestorov združuje obyvateľov bytov na Benediktiho ulici, číslo domov 1-5. Pri spoločných projektoch vzťahujúcich sa na všetky byty financujú sa z fondov spoločenstva. Na spustenie projektu treba súhlas 4/5 väčšiny spoločníkov. Keďže v dome na Benediktiho ulici 5 sa vlastníci rozhodli neparticipovať na tomto projekte pripojenia do Internetu[1] a budovanie domovej siete, bude sa realizovať tento projekt iba pre bytové domy Benediktiho 1 a 3. Reálny dôvod je – väčšina obyvateľov v dome na ulici Benediktiho 5 sú v dôchodkovom veku a možnosť mať v byte pripojenie do internetu nie je pre nich potrebná. V domoch na Benediktiho 1 a 3 je opačná situácia. Približne 90% obyvateľov vlastní počítač a väčšina z nich má pripojenie do internetu.

Zmysel vybudovať lokálnu počítačovú sieť s pripojením na Internet má iba vtedy, ak náklady na vybudovanie siete a cena za sprostredkovanie pripojenia je z dlhodobého hľadiska nižšia, než suma nákladov jednotlivých užívateľov na sprostredkovanie Internetu samostatne.

Obytné domy na Benediktiho ulici 1 a 3 sa nachádzajú v centre Bratislavy a sú situované pri budovách Slovenského rozhlasu (pyramída) a nového sídla Národnej banky Slovenska.
V budove číslo jedna sa nachádza 13 bytových jednotiek a v budove číslo 3 je 12.

Výhodou týchto budov, je pomerne malá vzdialenosť a vizuálny kontakt na výškovú budovu Slovenskej technickej univerzity. V budove STU sa nachádza stredisko SIX (The Slovak Internet eXchange) – Chrbticové pripojenie slovenského Internetu do sveta. V tomto stredisku majú umiestnené routre a servre veľký „chráči" na poli sprostredkovania Internetu. Vďaka týmto faktorom sa uvažuje aj o možnosti bez drátového pripojenia lokálnej počítačovej siete.

[1] Medzinárodná sieť. Internet je decentralizovaná štruktúra, založená na spojení medzi rozličnými počítačmi použitím jednoduchej technológie, t.j. (internetového protokolu IP). Na rozdiel od online služieb táto svetová sieť nemá oficiálneho správcu. Vďaka svojej decentralizovanej štruktúre je Internet nekontrolované médium. Internet bol do začiatku deväťdesiatych rokov 20. stor. relatívne neznámym pojmom. Jeho popularita vzrástla, keď sa do povedomia ľudí dostal World Wide Web (WWW). World Wide Web, ktorý si verejnosť často zamieňa s Internetom, je len jedna z množstva disponibilných služieb. Medzi ďalšie patrí e-mail, FTP, Gopher a IRC.

1.2 Stanovenie požiadaviek

Boli definované nasledujúce požiadavky:

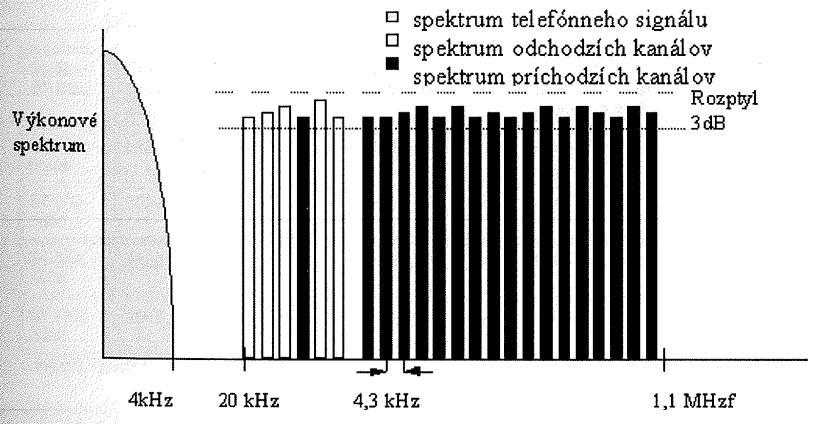
- Mojou úlohou je vybrať vhodný typ pripojenia na Internet. S najlepším pomerom ceny za sprostredkovanie pripojenia, kvality spojenia a rýchlosti pripojenia
- Navrhnúť a vybudovať fyzické zapojenie siete
- Zabezpečiť ochranu LAN v rámci pred útokmi z vonku
- Vybrať vhodné aktívne prvky lokálnej siete
- Zabezpečiť spravodlivé rozdeľovanie prenosovej šírky pripojenia
- Otestovať a doladiť kofiguráciu Rootra

ADSL	1,5 - 9 Mb/s, 16 -640 kb/s	Asymetrický	prístupové siete, internet, VOD
VDSL	13 - 52,8 Mb/s 1,5 -2,3 Mb/s	Asymetrický	prístupové siete, internet, VOD, ATM prístupová sieť
VDSL	36,8 Mb/s	Duplexný	prístupové siete, internet, VOD, ATM prístupová sieť

ADSL - Asymetrical Digital Subscriber Line je najznámejšia technológia.

Má odlišnú rýchlosť v príchodzom a odchodzom smere – je asymetrická. V príchodzom s rýchlosťami od 1,5 Mb/s do 9 Mb/s. V odchodzom smere od 16 kb/s do 640 kb/s. Maximálna vzdialenosť je 3 až 5 km

ADSL používa moduláciu DMT – metóda kódovania digitálneho signálu. Diskrétna multitónová modulácia DMT rozdelí frekvenčné spektrum do 256 oddelených kanálov (každý 4,3125 kHz s rýchlosťou po 60 kb/s). Dáta sú rozdelené na malé časti, sú pridelené do jednotlivých kanálov a je na ne aplikovaný algoritmus rýchlej fourierovej transformácie. Kanály 6 až 31 sa používajú pre smer od užívateľa a 32 až 250 pre smer k užívateľovi.



Obr 1. - ADSL prenosové spektrum

Cenník aDSL FLAT- Pevný mesačný paušál

- modem za 1 Sk, router za 1 Sk, niekoľkokrát vyššia rýchlosť ako bežné dial-up pripojenie až 2048/256 kbps (download/upload), ďalší prístup do siete Internet prostredníctvom dial-up zadarmo

Tab. 1. - aDSL FLAT- Pevný mesačný paušál

Produkt	Rýchlosť	Fair Use Policy	Agregácia	Zariar. za 1 sk	Dial up zdarma	Mesačná cena v Sk bez DPH pri viazanosti:			Cena za každý MB nad limit
						1	12	24	
FLAT Home	512/128	Áno	1:40	Modem*	Áno	499 Sk	399 Sk		-
FLAT Basic	1024/128	Áno	1:40	Router*	Áno	899 Sk	809 Sk	699 Sk	-
FLAT Business	1536/192	Áno	1:20	Nie	Áno	1 999 Sk	1 799 Sk	1 599 Sk	-
FLAT Profi	2048/256	Áno	1:20	Nie	Áno	3 099 Sk	2 789 Sk	2 499 Sk	-

Prameň: GlobalTel, a.s., Cenník služieb - http://www.globaltel.sk/index.php?option=com_content&task=view&id=18&Itemid=32, (10. 07. 2005)

*Zariadenie Modem/Router získa zákazník za 1 Sk pri podpísaní viazanosti 24 mesiacov.
- Predpokladom pre zriadenie služby je zriadenie a využívanie služby ST DSL od Slovak Telecom.
- Po uplynutí viazanosti zariadenia ostávajú vo vlastníctve spoločnosti GlobalTel a.s.

Tab. 2. - Inštalačné a mesačné poplatky účtované spoločnosťou Slovak Telecom za prípojku DSL:

Služba ST DSL	Dostupná rýchlosť v kbit/s (prijímanie / odosielanie dát)	Štandardná cena v Sk /mes.	Akciová cena - platná prvých 12 mesiacov od zriadenia služby	Zriadenie služby pri 24 mes. viazanosti
ST DSL doma	512 / 128	399,- Sk	249,- Sk	1,- Sk
ST DSL 1000	1024 / 128	699,- Sk	499,- Sk	999,- Sk
ST DSL 1500	1536 / 192	1099,- Sk	699,- Sk	999,- Sk
ST DSL 2000	2048 / 256	1799,- Sk	999,- Sk	999,- Sk

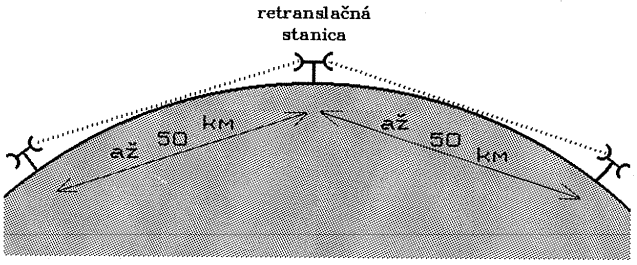
Prameň: GlobalTel, a.s. - http://www.globaltel.sk/index.php?option=com_content&task=view&id=18&Itemid=32, (10. 07. 2005)

2.2 Mikrovlnné pripojenie LAN do siete Internetu

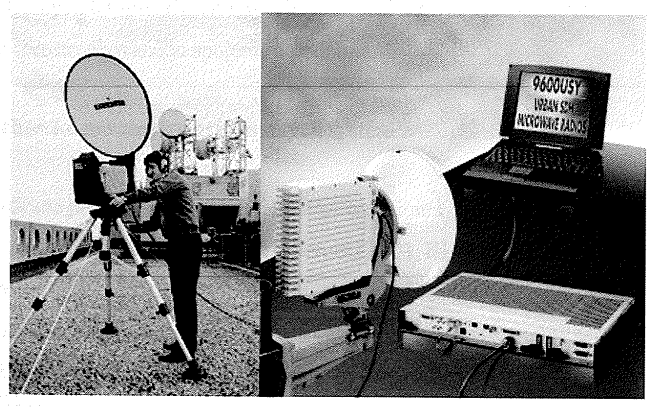
Rozľahlé počítačové siete využívajú k prepojeniu svojich uzlových počítačov najčastejšie pevné okruhy, prenajaté od spojovacích organizácií. Tieto okruhy sú obvykle vytvárané pomocou drôtových prenosových ciest (rôznych diaľkových káblov apod.). Existujú však aj iné možnosti (okrem verejných dátových sietí) - napríklad mikrovlnné.

Mikrovlnné spoje

Prívlastkom **mikrovlnné (microwave)** sa označujú elektromagnetické vlny s extrémne krátkou vlnovou dĺžkou resp. veľkou frekvenciou, ktorá je vlnovej dĺžke nepriamo úmerná. V praxi sa používa frekvencia od 1 do 12 GHz (tj. s vlnovou dĺžkou približne 30 až 2,5 cm). Vlny s takouto vysokou frekvenciou už možno, pomocou vhodných parabolických vysielacích antén, sústrediť do úzkeho lúča, a ten nasmerovať na prijímajúcu anténu. Úzko sústredený lúč vykazuje minimálny rozptyl, dovoľuje používať relatívne malý výkon vysielača a je veľmi odolný voči rušeniu. Na nižších frekvenciách nieje možné dosiahnuť potrebné sústredenie lúča, a na vyšších frekvenciách sa už začína znateľne prejavovať nepriaznivý vplyv atmosférických javov, ako napr. hmly a daždi.



Obr 2. - Nákres limitnej vzdialenosti mikrovlnných vysielačov



Obr 3. - Mikrovlnné spoje v praxi

Vzhľadom k priamočiaremu šíreniu sústredeného lúča elektromagnetických vĺn je dosah mikrovlnných spojov obmedzený na priamu viditeľnosť vysielača a prijímača. Tá je určovaná

konkrétnymi geografickými podmienkami a samozrejme tiež zakrivením Zeme. Možno ju umelo predlžovať umiesťovaním vysielacích a prijímacích antén na čo najvyššie veže. V rovine, kde sa uplatňuje iba vplyv zakrivenia zemského povrchu, je obvyklý dosah okolo 50 km. Pre preklenutí väčších vzdialeností je nutné budovať sieť retranslačných staníc - viď obr. Dosažiteľná prenosová rýchlosť na mikrovlnných spojoch je závislá na použitom frekvenčnom pásme a možnostiach prijímača a vysielača. Môže dosahovať hodnôt až 10 Mbit/sekundu.

Pre počítačové siete môžu byť mikrovlnné spoje výhodné napríklad v mestských aglomeráciách v tých miestach, kde neexistujú vhodné drôtové prenosové cesty a inštalácia nových neprichádza do úvahy (napr. v historických jadrách miest).

FWA[2]

FWA predstavuje prenosové zariadenie, ktoré slúži na realizáciu prístupových "Last mile" riešení poskytovateľa dátových služieb. Technológia pracuje na základe bezdrôtového prepojenia základňovej stanice s viacerými koncovými stanicami (terminálmi) umiestnenými na strane zákazníka.

Výhody použitia rádiových technológií spočívajú predovšetkým v jednoduchej implementácii do existujúcich systémov a vysokej flexibilite. Na základe daných vlastností je možné veľmi rýchlo reagovať na nové a meniace sa požiadavky zákazníka, napr. inštaláciu ďalších základňových staníc a zákazníckych terminálov, zmeny trás prepojení a zmeny prenosových rýchlostí, poskytovanie sprostredkovaného spojenia a podobne.

Základné technické parametre prístupových systémov

- Podporované prenosové rýchlosti 64kbit/s - nx2Mbit/s
- Podporované užívateľské rozhrania Ethernet,G.703, E1, T1, E3, V.35, X.21, AB linka
- Dostupnosť FWA riešenia 99,995 %
- Zabezpečenia FWA
- Štandardne Identifikácia účastníka cez sieťový manažment

[2] (Fixed Wireless Access) – FWA (Fixed Wireless Access) je širokopásmové bezdrôtové riešenie lokálneho prístupového okruhu. Označuje sa často aj ako Fixed Radio Wireless alebo Wireless in Local Loop. Prenos digitálnej informácie prostredníctvom rádiového signálu - či už ide o internet, prenos dát alebo hlasu v rámci mestských sietí - je novou zaujímavou alternatívou voči bežne používaným prístupovým technológiám (pevné linky, klasické metalické vedenie alebo optické vlákna), ktoré prepájajú koncového zákazníka s telekomunikačnou sieťou. Nastupuje tam, kde v miestnej prístupovej sieti nie sú k dispozícii pevné prístupové linky v požadovanej kvalite.

- Možnosť výberu SIM karta a šifrovací podsystém
- FWA pokrytie 3 - 5 km od základňovej stanice

2.2.1 Cenník mikrovlnného pripojenia

Pri 18 mesačnej zmluvnej viazanosti:

Tab. 3. - Cenník mikrovlnného spojenia

Typ MW spojenia	Šírka pásma	cena
Verejné pásmo 5,6 GHz	2048 Bit/s	10 400,- Sk
Poplatok za zriadenie spojenia		15 000,- Sk
Verejné pásmo 2,4 GHz	2048 Bit/s	9 700,- Sk
Poplatok za zriadenie spojenia		0,- Sk

Prameň: SWAN, a.s. – Slovak Wireless Access Network

2.3 Aktívny sieťový prvok - Rooter/firewall

Pri výbere tohto zariadenia sú najväčšie cenové rozdiely. Počnúc starým PC (napr. Pentium 233Mhz) s nainštalovaným operačným systémom Linux a komletným freeware[3] vybavením za rádovo stovky korún až po inteligentné manažovatelné routre alobo firewally in výrobcu Cisco, kde komponenty vhodné zhruba na našu sieť sa pohybujú približne:

- **Firewall** - 35 000,- Sk + nákup programového vybavenia + poplatky za technickú podporu
- **Rooter** - 50 000,- Sk + nákup programového vybavenia + poplatky za technickú podporu

Je zrejmé, že my sa vyberieme cestou najnižších nákladov, a vyberieme PC s primeranou konfiguráciou na to, aby jeho výkon postačoval na filtrovanie a reguláciu toku dát pod OS Linux

[3] Open source

Ide o softvér s voľne dostupnými zdrojovými kódmi. Tento pojem však nedefinuje len dostupnosť zdrojových kódov softvéru, ale aj ďalšie aspekty vývoja. Ide o celkový prístup k vývoju softvéru a k otázkam komerčného charakteru spojených s predajom a distribúciou softvéru. Azda najznámejším open source je operačný systém Linux.

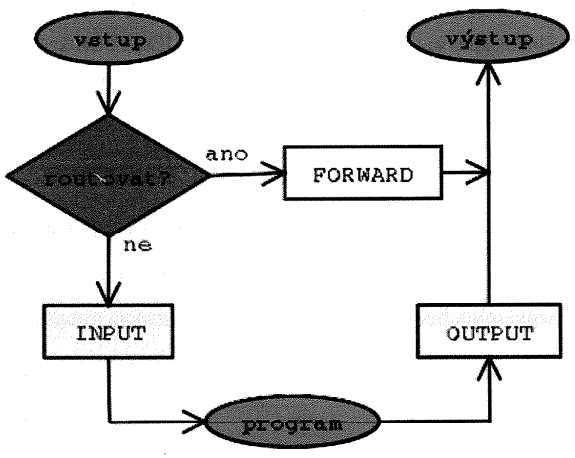
2.4 Softwérové vybavenie

Tak isto ako v predchádzajúcom bode môžeme ušetriť prostriedky aj tým, že operačný systém a programové vybavenie bude použité také, ktoré je voľne dostupné ZDARMA a zároveň spĺňa naše požiadavky na obsluhu vybraného hardwaru. Nakoľko najrozšírenejšie operačné systémy firmy Microsoft je možné použiť iba s kúpenou licenciou – použijeme programové vybavenie a OS Linux. Okrem iných výhod systému Linux voči systému Microsoft Windows je tá, že vyžaduje na prácu podobných úkonov menej systémových prostriedkov počítača.

3 PRINCÍPY BEZPEČNOSTI POČÍTAČOVEJ SIETE – FIREWALL

Máme linuxový router s dvoma sieťovými kartami (eth0 a th1), pričom na jednej strane je Internet a na druhej je lokálna sieť. Táto konfigurácia je v prostredí malých firiem veľmi častá, často býva na routeri prevádzkovaný aj www server a ďalšie služby, ktoré sú prístupné z internetu. Z takej situácie budeme vychádzať i my.

Každý IP datagram okrem vlastných užitočných dát so sebou nesie hlavičku, ktorá obsahuje IP adresu odosielateľa i adresáta, zdrojový a cieľový port, špecifikujúci program, ktorému je datagram určený a tiež ďalšie informácie popisujúce komunikáciu, ku ktorej datagram patrí. Paketový firewall je potom akýmsi filtrom, ktorý na základe týchto informácií rozhoduje o tom, ktoré pakety môžu byť pustené až k programom alebo naopak, ktoré môžu opustiť počítač.



Obr 4. - Schéma algoritmu pri filtrovaní paketov

Každý paket, nech už je jeho pôvod akýkoľvek, prechádza systémom reťazcov (chainov) ktoré tvoria tzv. filtrovaciu tabuľku, ako to je vidieť na priloženom obrázku. Pre ozrejmenie si môžeme predstaviť paketový filter ako potrubie jednotlivé reťazce ako ventily prepúšťajúce len vybrane pakety

- na začiatku sa snaží jadro rozhodnúť, či je prichádzajúci paket určený pre tento počítač alebo či je potreba routovať ho inam.

- ak je adresátom on sám, predá paket k ďalšiemu spracovaniu do vstupného /INPUT/ reťazca. Ak vyhovie filtrovacím pravidlám, dostane ho k spracovaniu niektorý z lokálnych programov na cieľovom porte
- ak je datagram určený niekomu inému a je zkonfigurovaný ako router, teda ak je povolené routovanie paketov premennou /proc/sys/net/ipv4/ip_forward == 1, paket bude postúpený do reťazca FORWARD a počítač sa ich potom pokúsi podľa svojich možností doručiť ich príjemcovi. Ak je smerovanie paketov zakázané /implicitný stav/, bude paket zahodený.
- poslednou možnosťou je, že datagram vytvoril niektorý z lokálnych programov. Potom je nutné, aby paketový filter opustil cez reťazec OUTPUT.

Okrem spomenutých reťazcov INPUT, OUTPUT a FORWARD existujú ešte ďalšie dva, a to PREROUTING a POSTROUTING, ktoré sa predovšetkým používajú na iné účely ako v filtrovaniu paketov. Ako je zrejmé z názvov, PREROUTING je aktívny v dobe pred routovaním, t.j. prechádzajú ním pakety určené pre lokálny stroj, tiež pakety, ktoré budú nasmerované inam. Tiež POSTROUTINGem pretekajúce pakety odchádzajú z počítača rovnako ako smerované datagramy. Tieto reťazce majú zvláštny význam pri preklade adries (NAT)

IPtables

K nastavovaniu pravidiel slúži nástroj Iptables, ktorý je súčasťou snáď všetkých nových distribúcií. Jeho použitie je viazané na jadrá verzií 2.4. Aj keď je možné na týchto verziách používať tiež starší program ipchains, nemôžeme to s čistým svedomím odporučiť, pretože podpora ipchains na nových kerneloch je iba prechodnou záležitosťou a navyše sa tým pripravujete o niektoré nové vlastnosti iptables.

Použitie iptables možno pôsobí na prvý pohľad zložito, ale v skutočnosti sú veľmi logické a intuitívne. Program voláme s niekoľkými parametrami. Prvým je miesto určenia, kam chceme pravidlo zaradiť. Ak chceme napríklad nejaké pravidlo pridať /append/ do reťazca INPUT, urobíme to asi takto:

```
iptables -A INPUT pravidlo
```

pravidlo je určené špecifikáciou, s ktorou sa skúmaný paket porovnáva. napríklad zápisu:

```
-p TCP -i eth0 -s 192.168.0.2 --dport 80
```

vyhovujú všetky IP datagramy, ktoré boli prijaté sieťovou kartou eth0, majú ako odosielateľa uvedenú stanicu 192.168.0.2 a sú určené pre ľubovoľného príjemcu a jeho TCP port číslo 80. Všimneme si, že ak niektorý parameter explicitne neuvedieme, bude pravidlo vyhovovať ľubovoľnému parametru z množiny možných. Ak teda napríklad neurčíme cieľovú IP adresu pomocou "-d", výstupnou hodnotou bude 0/0, teda ľubovoľná IP adresa

Ďalším parametrom býva "-j", ktorým určujeme, čo má jadro urobiť, ak skúmaný paket danému pravidlu vyhovuje. Možností ako s ním naložiť je celá rada. Najpoužívanejším cieľom (target) býva DRP (paket bude zahodený) alebo ACCEPT (paket bude pustený). Ďalším cieľom môže byť napríklad REJECT (paket bude zahodený, ale jeho pôvodca bude o tom informovaný, pomocou chybového hlásenia ICMP) alebo LOG (záhlavie paketuje zapísané do systémového logu)

Ukážka kompletnej definície pravidla:

```
iptables -A OUTPUT -i eth1 -p TCP -s 192.168.0.1 --sport 3000 \
   --d 192.168.0.2 --dport 25 -j DROP
```

Preložené: Ak sa v reťazci OUTPUT objaví TCP segment, ktorý bude mať v úmysle opustiť počítač cez rozhranie eth1, jeho odosielateľom bude 192.168.0.1 port 3000 a príjemcom port 25 adresy 192.168.0.2, ta ho nepúšťaj.

Syntax iptables môžu byť veľmi rôznorodé a spôsobov, ako špecifikovať množinu datagramov môže byť mnoho.[4]

3.1 Výber a nastavenie filtrovania paketov

Otázka by skôr mala znieť: Aké pakety prepúšťať? Firewall je totiž vhodné koncipovať v zmysle vety „čo nie je vyslovene povolené, je zakázané" Znamená to, že implicitnou politikou vo všetkých základných reťazcoch by malo byť zahadzovanie všetkých paketov, ktoré nevyhovujú

[4] Podrobný popis parametrov iptables môžeme nájsť okrem manuálovej stránky iptables (8) tiež v Linux 2.4 Packet Filtering HOWTO - http://netfilter.samba.org/unreliable-guides/

niektorému z pravidiel. Opačným prístupom by bolo povoliť všetko a filtrovať iba nežiaduce toky. My však dodržiavame prvý spôsob, ktorý býva bezpečnejší.

Začneme teda zápisom:

```
iptables -P INPUT DROP
iptables -P OUTPUT DROP
iptables -P FORWARD DROP
```

Pakety vo vstupnom (INPUT) reťazci podľa cieľového portu, ktorému je paket určený. Ak má byť z na našom počítači zo strany internetu dostupný www a SMPT server, povolia sa tieto služby takto:

```
iptables -N tcp_segmenty
iptables -A INPUT -p TCP -i eth0 -j tcp_segmenty
iptables -A tcp_segmenty -p TCP --dport 80 -j ACCEPT
iptables -A tcp_segmenty -p TCP --dport 25 -j ACCEPT
```

Význam tohto zápisu je nasledovný: Vytvorí (-N) sa nový reťazec, ktorý bude spracovávať všetky TCP segmenty prichádzajúce cez rozhranie eth0. Do tohto reťazca pridáme (-A) pravidla, ktoré stanovujú, že a príde segment adresovaný portu 80 (HTTP) alebo 25 (SMTP), prepustia ho. Podobne by mohol vyzerať zápis pre služby používajúce UDP:

```
iptables -N udp_pakety
iptables -A INPUT -p UDP -i eth0 -j udp_pakety
iptables -A udp_pakety -p UDP --dport 53 -j ACCEPT
```

Všeobecne nie je nutné vytvárať nové reťazce a pravidlá sa môžu písať priamo d materského INPUTu, ale ak si dáme prácu a usporiadame pravidlá, ktoré spolu súvisia, budeme odmenení vyššou prehľadnosťou. Neskôr si ukážeme ďalšie užitočné využitia takéhoto spôsobu zápisu

Zvláštnym prípadom datagramov sú ICMO, teda servisné pakety používané na prenos diagnostických a chybových správ. Pre správne fungovanie je potrebné prepúšťať prinajmenšom ICMP typ 3 – "destination unreachable". Vhodné je povoliť tiež 0 – "Echo reply", 8 – "Echo request" a 11 – " Time exceeded", ktoré používajú užitočné programy ping a traceroute. Ostatné ICMP správy môžete s čistým svedomím filtrovať.

```
iptables -A INPUT -p ICMP -i eth0 --icmp-type 0 -j ACCEPT
```

```
iptables -A INPUT -p ICMP -i eth0 --icmp-type 3 -j ACCEPT
iptables -A INPUT -p ICMP -i eth0 --icmp-type 8 -j ACCEPT
iptables -A INPUT -p ICMP -i eth0 --icmp-type 11 -j ACCEPT
```

Ak máme počítač pripojený tiež do LAN alebo inej dôveryhodnej siete, môžeme povoliť pakety prichádzajúce cez rozhranie eth1, rovnako ako cez systémové (loopback) rozhranie lo.

```
iptables -A INPUT -p ALL -i eth1 -j ACCEPT
iptables -A INPUT -p ALL -i lo -j ACCEPT
```

Všetky ostatné prichádzajúce datagramy sú v súlade s implicitnou politikou zamietnuté, preto ich budeme logovať, aby sme podľa nich mohli diagnostikovať prípadné problémy.

```
iptables -A INPUT -j LOG
```

U jednoduchších firewallov sa nemusíme zaoberať nejakým zložitým filtrovaním v reťazci OUTPUT, pretože datagramy, ktoré tade prechádzajú, majú svoj pôvod v našom počítači, preto nepredstavujú nejaké zvlášť bezpečnostné riziko. Pre začiatok bude stačiť niečo takéto.

```
iptables -A OUTPUT -p ALL -s 127.0.0.1 -j ACCEPT
iptables
-A OUTPUT -p ALL -s 192.168.0.1 -j ACCEPT
iptables -A OUTPUT -p ALL -s 1.2.3.4 -j ACCEPT
iptables -A OUTPUT -j LOG
```

Tým sme prepustili všetky pakety, ktorého odosielateľom je náš počítač. (Adresa 127.0.0.1 je adresou loopbacku, adresa 1.2.3.4 tu zastupuje našu verejnú IP adresu, a nakoniec predpokladajme, že 192.168.0.1 je adresa nášho routeru na miestnu LAN. Prípadné ostatné pakety budú zahodené a logované.

Reťazec FORWARD ma svoj význam v prípade, keď máme v počítači viac sieťových rozhraní a chceme smerovať pakety z jednej siete do druhej, Ak má byť náš firewall routerom medzi internetom dostupným z rozhrania eth0 a vnútornej siete na eth1, použijeme nasledovný zápis:

```
iptables -A FORWARD -i eth1 -j ACCEPT
iptables -A FORWARD -i eth0 -o eth1 -m state \
  --state ESTABLISHED,RELATED -j ACCEPT
```

Povolíme ním neobmedzené smerovanie paketov z vnútornej siete. Pakety v opačnom smere bude prepustené iba v prípade, že patria k nejakému už existujúcemu spojeniu. Syntaxou „–m state“ si zatiaľ nemusíte príliš lámať hlavu, vedzte, že je určená stavovému firewallu, o ňom detailnejšie neskôr.

Ak je jednou zo sietí Internet a druhou LAN so súkromnými IP adresami, ktoré sa nedajú používať s vonkajším svetom, musíme skonfigurovať ešte SNAT

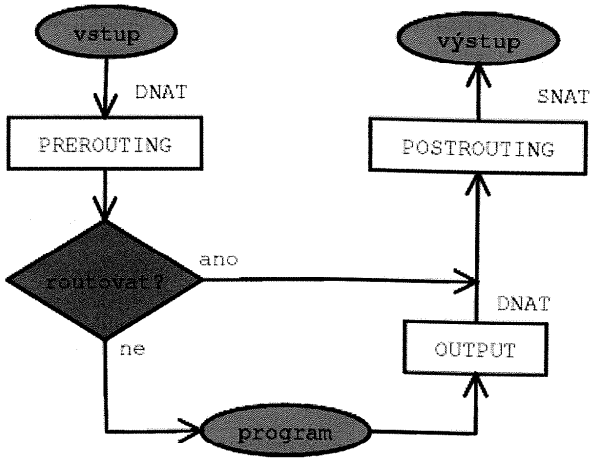
```
iptables -t nat -A POSTROUTING -o eth0 -j MASQUERADE
```

Zápis znamená, že router bude pakety odchádzajúce cez rozhranie eth0 „maskovať“ svojou IP adresou (1.2.3.4). Zvonku to bude vyzerať tak, že všetky pakety budú cez Internet putovať s IP adresou 1.2.3.4 a až na našom firewalle sa budú prekladať na adresy vnútornej siete.

3.2 NAT – Network Adress Translation

Okrem filtrovanej tabuľky, ktorej fungovanie sme si už ozrejmili, má linuxové jadro ešte ďalšiu tabuľku, ktorou prechádzajú pakety a to NAT (Network Adresa Translation) Rovnako ako filtrovacia tabuľka, aj ona obsahuje tri reťazce, ktoré sa už však nepoužívajú k filtrovaniu paketov (aj keď aj to je možné), ale ako už jej názov napovedá, k zmenám adries datagramov. Funguje to tak, že ak paket pri priechode vyhovie zadanému pravidlu, tak je mu podľa určeného vzoru zmenená adresa jeho odosielateľa, resp. príjemcu podľa určeného vzoru. Podľa toho hovoríme buď o preklade adries (SNAT – Source NAT) alebo o preklade adries príjemcu (DNAT – Destination NAT)

Podobne ako sme si ukázali štruktúru filtrovacej tabuľky, naznačíme si ako vyzerá púť datagramov cez NAT tabuľku



Obr 5. - Prechod datagramov cez NAT tabulku

- Prichádzajúce pakety, nech už je ich destinácia akákoľvek, prechádza reťazcom PREROUTING, kde im môžeme meniť adresu príjemcu, teda DNAT
- Odchádzajúce pakety, bez ohľadu na odosielateľa zase môžu byť SNATované (zamaskované adresy odosielateľa) v reťazci POSTROUTING
- Zvláštnym prípadom je odchádzajúci reťazec OUTPUT, ktorý je možno použiť na DNATovanie paketov vzniknutých iba na lokálnom počítači. V praxi sa však OUTPUT DNAT používa iba zriedka.

3.3 SNAT

Príkladom použitia SNATu je IP maškaráda, teda zamaskovanie IP adresy routovaných paketov adresou routeru. Znie to komplikovane, je to veľmi jednoduché. Predstavme si, že máme od poskytovateľa pripojenia k dispozícii jednu verejnú IP adresu, ale potrebujeme k sieti pripojiť veľké množstvo počítačov. Preto verejnú adresu pridelíme routeru, ostatným počítačom pridelíme súkromné /privátne/ IP adresy, ktoré sú k tomuto účelu rezervované podľa RFC xxxx a do NAT tabuľky routeru vložíme nasledovný príkaz:

```
iptables -t nat -A POSTROUTING -o eth0 -j SNAT --to 1.2.3.4
```

Príkaz spôsobí, že ak zo siete (napr. z adresy 192.168.0.2) príde paket, ktorý má v úmysle opustiť router cez vonkajšie rozhranie eth0, tam dôjde k nahradeniu jeho pôvodnej IP adresy za adresu

1.2.3.4. To znamená, že príjemca datagramu bude mať „pocit" ako keby s ním nekomunikovala stanica s adresou 192.168.0.2 ale priamo router 1.2.3.4, ktorý ale zaistí, aby sa pakety v opačnom smere dostali naspäť k 192.168.0.2.

Na uvedenom príkaze je dôležité určenie „-t nat", čo znamená, že pravidlo je určené k zapísaniu do NAT tabuľky. Podobne by sme mohli pravidlá do filtrovacej tabuľky uvádzať príkazom „-t filter", ale nie je to nutné, pretože „-t filter" je implicitný príkaz. Už sme si uviedli iný príklad definície maškarády:

```
iptables -t nat -A POSTROUTING -o eth0 -j MASQUERADE
```

Obidva prístupy fungujú takmer rovnako, ale pri druhom spôsobe nemusíme pri deklarácii uvádzať maskovanú adresu. To je výhodné vtedy, keď ju v okamihu zavedenia pravidla ešte nepoznáme, napríklad ak ešte len získame z DHCP servera. Prvý spôsob naopak prináša možnosť špecifikovať nie jednu, ale množstvo adries, vtedy ak má náš router k dispozícii niekoľko verejných adries.

```
iptables -t nat -A POSTROUTING -o eth0 -j SNAT --to 1.2.3.4-1.2.3.8
```

3.4 DNAT

Naopak DNAT použijeme vtedy, ak potrebujem meniť IP adresy adresáta datagramu. Používa sa to pri presmerovaní datagramov na iného adresáta. V dobách kernelov 2.0 a 2.2 sme namiesto DNAT-" používali rôzne špeciálne programy na presmerovanie portov, napríklad ipmasqadm, ipfwd, portfwd a pod. Vhodný je častý príklad, keď máme niekde v sieti umiestnený www server a je potrebné na neho presmerovať všetky požiadavky smerujúce cez náš router:

```
iptables -t nat -A PREROUTING -p tcp --dport 80 -i eth0 \
  -j DNAT --to 192.168.0.2:80
```

Týmto zápisom je zariadené, že ak sa na našom routeri objaví TCP segment určený pre port 80 (www server), tak sa jeho cieľová adresa prepíše na 192 .168.0.2. i.

Tým čím je „-jMASQUARADE" pre SNAT je výraz „-j REDIRECT" pre DNAT, teda dáva nám možnosť presmerovať datagram na iný port bez znalosti presnej IP adresy. Ak chceme

napríklad použiť program SQUID na porte 3128 v režime transparentného proxy, presmerujeme všetky HTTP požiadavky na port 3128.

```
iptables -t nat -A PREROUTING -p tcp --dport 80 -i eth1 \
  -j REDIRECT --to-port 3128
```

Je potrebné nezabudnúť na skutočnosť, že ak nejaký paket vyhovel pravidlu v NAT tabuľke, neznamená to, že má plný priechod cez paketový filter. Vždy je treba ešte povoliť dané spojenie i vo filtrovacej tabuľke, hlavne ak ju mame inicializovanú pre implicitný zákaz (DROP). Napríklad pre vyššie uvedené presmerovanie www serveru DNATem by bolo potrebné:

```
iptables -A FORWARD -i eth0 -p tcp -d 192.168.0.2 --dport 80 \
  -m state --state NEW,ESTABLISHED,RELATED -j ACCEPT
```

Rovnako tak pre správne fungovanie maškarády (SNATu) je nutné uvoľniť smerovanie paketov a to oboma smermi:

```
iptables -A FORWARD -i eth0 -o eth1 -m state \
  --state ESTABLISHED,RELATED -k ACCEPT
iptables -A FORWARD -i eth0 -o eth1 -j ACCEPT
```

Rozdiel medzi filtrovacou tabuľkou a NAT tabuľkou je tiež v tom, že zatiaľ čo filtrovacou tabuľkou prechádza každý paket bez rozdielu, tak NAT tabuľkou je aktívny iba pre prvý paket nového spojenia. Preto teda nie je vhodné používať NAT tabuľku pre nejaké zložité filtrovanie a keď už, tak iba s podporou stavového firewallu. S výhodou môžeme zahadzovanie paketov v NAT tabuľke použiť ako ochranu pred IP spoofingom z nezmyselných IP adries, proste preto, že pravidlá stačí raz napísať do PREROUTINGu a nemusíme ich zdvojovať v INPUTe a FORWARDe:

```
iptables -t nat -A PREROUTING -i eth0 -s 192.168.0.0/16 -j DROP
iptables -t nat -A PREROUTING -i eth0 -s 172.16.0.0/12 -j DROP
iptables -t nat -A PREROUTING -i eth0 -s 10.0.0.0/8 -j DROP
```

3.5 Stavový firewall

Jadrá rady 2.4 prichádzajú s koncepčne novým prístupom, ked pri filtrovaní berú do úvahy nielen informácie obsiahnuté v hlavičke skúmaného datagramu, ale tiež dokážu na neho pozerať komplexne, v kontexte spojenia, do ktorého patria. Stavový firewall rozozná paket, ktorý otvára

nové spojenie od paketov, ktoré túto komunikáciu realizujú, a vďaka tomu môžeme filtrovať dátové toky precíznejšie.

Každý skúmaný datagram (a to nielen TCP segment, ale i UDP paket) je potom zaradený do niektorej z týchto kategórii:

- NEW – datagram otvára novú komunikáciu
- ESTABLISHED, RELATED - datagram je súčasťou už naviazaného spojenia alebo s ním nejakým spôsobom súvisí
- INVALID - datagram nie je súčasťou žiadneho spojenia alebo sa ho nepodarilo identifikovať

Na základe stavovej informácie môžeme teda pakety triediť. Môžeme napríklad stanoviť, že spojenia môžu byť nadväzované iba smerom z vnútornej siete von a nie naopak, pričom pakety už otvorených spojení môžu putovať oboma smermi.

```
iptables -P FORWARD DROP
iptables -A FORWARD -i eth0 -m state --state ESTABLISHED,RELATED \
  -j ACCEPT
iptables -A FORWARD -i eth1 -j ACCEPT
```

3.5.1 Výhody stavového firewallu

Stavový firewall prináša značné zjednodušenie filtrovacích pravidiel oproti dobám, ked sme pomocou ipchains museli brať do úvahy rôzne kombinácie adries, vysokých portov a SYN flagov. Teraz môžeme, vďaka klasifikácie paketov, implicitne povoliť, aby firewallom prechádzali pakety k už naviazaným spojeniam (ESTABLISHED) a obmedzenie stanovujeme na úrovni naviazaného spojenia.

Nová inteligencia stavového firewallu dovoľuje prísnejšie „lustrovanie" paketov. Teraz už môžeme odstrániť aj podvrhnuté pakety, ktoré sa statickým firewallom javili ako nezávadné, ale v kontexte prebiehajúcich spojeniach ich stavový firewall dokáže odhaliť. Vďaka tomu sa môžeme lepšie brániť pred rôznymi technikami fingerprintingu (zisťovanie typu OS) a portscanningu (zisťovanie otvorených portov), ktoré môže útočník zneužiť pri odhaľovaní slabín našej siete.

Stavový firewall rieši s konečnou platnosťou fungovanie pasívneho a hlavne aktívneho FTP režimu (vid nižšie), čo bolo predtým prinajmenšom problematickou záležitosťou

Nevýhody stavového firewallu

V porovnaní s výhodami zanedbateľné. Jedinou, ktorá mi napadá sú vyššie nároky na hardware firewallu. V pamäti je totiž nevyhnutné udržovať stavové informácie o všetkých spojeniach.

3.6 Aktívne a pasívne FTP

Známym dlhodobým problémom pri tvorbe firewallových pravidiel je FTP. FTP môže fungovať v dvoch režimoch – pasívnom a aktívnom. V aktívnom režime klient odošle (na port 21) servera číslo potu (nad 1024) a server sa na neho zo svojho portu (20) pripojí. Pasívny režim funguje presne opačne – server pošle klientovi port (nad 1024) a on sa na neho pripojí (z › 1024). Novšie programy, hlavne www browsery, preferujú pasívny FTP režim, ktorý je považovaný za bezpečnejší, najviac vďaka tomu, že dátové spojenie je nadväzované v rovnakom smere ako pôvodná požiadavka.

Problémom FTP je, že pracovný port nie je statický, ale mení sa s každým spojením. Preto nie je ľahké (a ani možné) napísať bezpečné a jednoznačné statické filtrovanie pravidlo, ktoré by dokázalo regulárne FTP spojenie rozoznať.

Preto firewall potrebuje oporu v jadre, respektíve v module ip_conntrack_ftp, ktorý dokáže z nadväzovaného spojenia „odpočúvať" dohodnutú kombináciu portov a zaistí, že stavový firewall bude takéto pakety klasifikovať ako „RELATED" Ak chcete, aby váš firewall podporoval FTP prenosy, zaveďte spomínaný modul a povoľte prichádzajúce RELATED spojenie.

```
modprobe ip_conntrack_ftp
iptables -A FORWARD -i eth0 -m state --state RELATED,ESTABLISHED -j
ACCEPT
```

Dodajme, že podobne ako FTP sa chová tiež IRC a niektoré ďalšie protokoly. Kernel 2.4 obsahuje podporu len pre FTP a IRC.

3.7 AUTH

Niektoré programy používajú službu AUTH (identd) na zistenie, akému užívateľovi patrí to ktoré spojenie. Pretože môže poskytovať útočníkovi nežiaduce údaje, nie je dobré otvárať ho

svetu. Ale ak ju bežným spôsobom zablokujeme (DROP), tiež to nie je dobre, pretože niektoré programy (napr. SMTP servery) ju môžu využívať v rámci užitočnej prevádzky. Ak ju teda budeme filtrovať, môže dochádzať k značnému spomaleniu komunikácie, keď bude protistrana čakať na vypršanie času vyhradeného k vybaveniu AUTH požiadavky.

Preto je nevyhnutné na firewallu blokovať AUTH nie pravidlom DROP, ale REJECTom. Rozdiel je taký, že zatiaľ čo DROP pakety doslova zahodí, REJECT ich „zdvorilo odmietne", t.j. vygeneruje odosielateľovi ICMP datagram s oznámením o odfiltrovaní dotyčného paketu.

```
iptables -A INPUT -i eth0 -p TCP --dport 113 -j REJECT
```

3.7.1 Ochrana pred IP spoofingem

IP spoofing znamená falšovanie zdrojovej IP adresy, čím sa útočník snaží predstierať, že je niekto iný, alebo sa snaží zamaskovať svoju pravú IP adresu. Preto je žiaduce, aby sa na vstupe firewallu filtrovali pakety, ktoré sú evidentne podvrhnuté a ich pôvodca nemôže mať čisté úmysly.

Najzákladnejšou ochranou pred IP spoofingem predstavuje rp_filter. Ide o jednoduchú ochranu, ktorá je viazaná na konkrétne sieťové rozhranie. Funguje tak, že jadro zablokuje pakety so zdrojovou adresou, ktorá by podľa routovacej tabuľky mala prísť z iného dostupného rozhrania. Príklad:

```
echo "1" > /proc/sys/net/ipv4/conf/eth0/rp_filter
```

alebo všeobecnejšie:

```
for interface in /proc/sys/net/ipv4/conf/*/rp_filter; do
      echo "1" > ${interface}
  done
```

Ak je adresa vnútornej siete napr. 192.168.0.0/24 jadro zaistí, že cez rozhranie eth0 neprenikne žiadny paket, ktorý by sa snažil tváriť, akoby mal svoj pôvod v tejto vnútornej sieti. Rovnako tak bude blokovať 127.0.0.0/8, pretože tieto adresy patria loopbacku...

rp_filter je jednoduchou, ale účinnou ochranou pred IP spoofingom. Ak nepoužívate nejaké zložité routovacie techniky, mali by ste ho mať zapnutý.

Rozšírenou ochranou pred IP spoofovaním riešime až na úrovni paketového filtra. RFC 1918 udáva rozsahy IP adries, ktoré sú vyhradené pre použitie v lokálnych sieťach a jeho datagramom nie je dovolené smerovať do internetu. Ak sa taký paket objaví na internetovej strane routeru, nemôže ísť o žiaduce spojenie a je treba ho odfiltrovať:

3.8 iptables -N spoofing

```
iptables -A spoofing -s 192.168.0.0/16 -j DROP
iptables -A spoofing -s 172.16.0.0/12 -j DROP
iptables -A spoofing -s 10.0.0.0/8 -j DROP

iptables -A INPUT -i eth0 -j spoofing
iptables -A FORWARD -i eth0 -j spoofing
```

V uvedenom príklade je najprv definovaný nový reťazec „spoofing" a prebiehajú ním prichádzajúce pakety z reťazcov INPUT a FORWARD. Týmto zápisom sa ušetrí duplikovanie zákazov na dvoch miestach.

Okrem adries rezervovaných v RFC 1918 existuj i ďalšie adresy, s ktorými by sme sa na internete nemali stretnúť. Ak je v stĺpci uvedené „reserved", nie je daný rozsah adries pridelený a môžeme ich filtrovať.Treba dať pozor, pretože rezervované adresy môžu byť v budúcnosti pridelené.

3.9 SYN flooding

Obľúbená crackerská technika, ktorou sa realizuje DoS (Denial of Service) útok. Spočíva v tom, že útočník zahltí cielový počítač množstvom TCP segmentov s nastaveným SYN flagom, ktoré požadujú vytvorenie nového spojenia. Prostriedky serveru sú samozrejme obmedzené, a preto skôr alebo neskôr dôjde k DoS.

Ochrana spočíva v stanovení limitu pre novo vytvorené spojenia (SYN pakety) v čase. Pomocou iptables môžeme vytvárať pravidlá, ktorými síce paket povolíme (ACCEPT), ale zároveň určujeme, že pravidlo smie byť aplikované nie častejšie než napríklad 4x za sekundu. Tak napríklad primeraná ochrana nášho serveru pred SYN floodingom môže byť:

```
iptables -N syn_flood
```

```
iptables -A INPUT -i eth0 -p tcp --syn -j syn_flood
iptables -A syn_flood -m limit --limit 1/s --limit-burst 5 -j RETURN
iptables -A syn_flood -j DROP
```

Preložené: Vytvorili sme nový reťazec, do ktorého predávame TCP pakety s flagom SYN prichádzajúce z eth0. Ak bude takých paketov menej než 5 za sekundu, vrátime ich späť do INPUTU. Inak ich filtrujeme.

Inteligentné logovanie

Logovanie zahodených paketov je vhodný nastroj pri diagnostikovaní problémov, ale každý, kto kedy prehľadával logy nejakého reštriktívne navrhnutého firewallu iste vie, že môžu narásť do niekedy až do nepraktických objemov. Preto sa snažíme zamedziť logovanie viacero rovnakých záznamov pomocou rovnakej techniky ako SYN floodingu. Namiesto obyčajného logovacieho pravidla napíšeme:

```
iptables -A INPUT -m limit --limit 3/hour --limit-burst 5 -j LOG
```

Uvedený zápis loguje iba prvých 5 paketov, ale nie častejšie ako 3x za hodinu

Ping of death

Ďalšia nepríjemná DoS technika spočíva v zavalení obete žiadosťami o echo-request (ping). Brániť sa pred ňou môžeme stanovením limitu na ICMP echo-request požiadavkami. Päť pingov postačí:

```
iptables -A INPUT -p icmp --icmp-type echo-request \
    -m limit --limit 1/s --limit-burst 5 -j ACCEPT
```

Tak a tým som s výkladom základov firewallovania na Linuxe 2.4 pri konci. Ak chcete vidieť ako vyzerá taká hotová konfigurácia, dovolil som si zhrnúť poznatky vysvetľované vo všetkých troch častiach tohto seriálu do príkladového skriptu, ktorá nájdete tu, ktorý si môžete ihneď začať používať alebo sa ním inšpirovať pri stavbe svojho vlastného firewallu.

4 PRINCÍP REGULÁCIE PRENOSU DÁT

QoS[5] (Quality of Services) služby

V klasické jednoduché síti typu "Internet" se všichni její uživatelé dělí o prostředky sítě stejným dílem - např. přenáší-li data sto lidí po lince s kapacitou 1Mb/s, přenáší se každému jeho data rychlostí 10kb/s. Není to problém, pokud aplikace pracuje při každé rychlosti a je nutné "jen" počkat na www stránku o něco déle.

Existují ale aplikace, které potřebují minimální zaručenou rychlost dat, jinak nefungují - například IP telefonie. V "klasické" síti může taková aplikace fungovat, dokud síť není zatížená. Jakmile ale zátěž vzroste (např. během hovoru si deset kolegů začne prohlížet www stránky), rychlost se sníží na takovou hodnotu, která není pro přenos hlasu postačující.

V síti, která podporuje QoS (Quality of Services), je možné pro takový případ potřebný tok dat rezervovat technikými prostředky a je tedy možné provozovat spolehlivě určité aplikace i při plném zatížení sítě.

Pasnet nabízí poskytování služeb QoS a to na úrovni IP a ATM.

- pozadie
- základy regulácie prenosu dát

4.1 Pozadie

Jeden zo základných drawbacks TCP/IP protokolu je nedostatok správnej QoS funkčnosti. QoS je schopnosť garantovať limity prenosovej šírky pre konkrétnu službu a užívateľa.

Hoci existujú protokoly ako DiffServ a ďalšie riešenia ktoré môžu ponúknuť QoS vo veľkých počítačových sieťach, žiadna z nich nedosiahne dostatočne vysoký štandard pre široké použitie.

Ďalšou skutočnosťou je, že väčšina súčasných QoS riešení je na báze aplikácií, tj., že pracujú ako aplikácie zastupujúce počítačovú sieť s QoS informáciami. Z pohľadu bezpečnosti je

[5] Prameň: http://www.pasnet.cz/sluzby/qos.html, 10.7.2005

samozrejme neakceptovateľné, aby aplikácia, (tj, užívatelia), rozhodovala o prioritách jej vlastného dátového prenosu v rámci počítačovej siete. Pri návrhoch bezpečnostných systémov, kde užívateľ je nedôveryhodný, sieťové vybavenie by malo rozhodovať o prioritách a nastaveniach šírok prenosu.

Tieto vymenované skutočnosti vysvetľujú, prečo je skoro nemožné nastaviť priority a garantovať limity prenosu dát vo veľkej a komplexnej počítačovej sieti, kde sú použité rôzne štandardy a produkty. Internet je dobrým príkladom takejto sieťovej štruktúry.

Na druhej strane, v dobre regulovaných sieťach existujú vynikajúce možnosti použitia rôznych metód na kontrolu prenosu. Dobre regulovaná sieť je zväčša definovaná v obmedzení správy a nie vo veľkosti siete. Dátový prenos v MAN[6] a dokonca aj vo veľmi veľkých WAN[7] môžu byť veľmi dobre spravované za predpokladu, že je sieť navrhnutá v homogénnej štruktúre.

Náš Firewall sám zabezpečuje funkčnosť QoS použitím obmedzení a garancií pre prenesené dáta na sieti radšej, ako by sa mal spoliehať a dôverovať aplikáciám a užívateľom. Teda preto je dobre nastavený na správu prenosu pre malé LAN[8].

4.2 Základy regulácie dátového prenosu

Najjednoduchší spôsob ako dosiahnuť QoS na sieti aj z hľadiska bezpečnosti ako aj z pohľadu závislosti na budúcom rozvoji siete je mať časti siete, a nie aplikácie, zodpovedné za kontrolu regulácie dát na sieti v dobre nastavených obmedzovacích bodoch.

Regulácia prenosu dát pracuje na princípe merania a zoraďovania IP paketov prepúšťané v závislosti na množstve konfigurovateľných parametrov. Diferencované prenosové obmedzenia a garancie v závislosti na zdroji a cieli a parametroch protokolu môžu byť vytvorené podobným spôsobom ako práva na Firewall-e. Regulácia dát pracuje na princípoch:

- Použitie prenosových obmedzení zoraďovaním paketov ktoré presahujú nastavené limity a ich posielanie neskôr, keď je vyťaženie nižšie.
- Vypúšťanie paketov ak zásobník na pakety je plný. Vypustený paket by mal byť vybraný z tých, ktoré sú zodpovedné za vyťaženie prenosovej šírky.

[6] Metropolitan Area Network – počítačová sieť zastrešujúca mestskú aglomeráciu
[7] Wide Area Network – počítačová sieť spájajúca dve a viac budov, v rámci mestskej aglomerácie
[8] Local Area Network – počítačová sieť vybudovaná v rámci budovy, spájajúca jednotlivé poschodia

- Nastavenie prenosu v závislosti na nastavení administrátora – ak prenos s vyššou prioritou rastie pokiaľ komunikačná linka je plná, prenosy s nižšou prioritou by mali byť dočasne obmedzené na zabezpečenie priestoru pre prenosy s vysokou prioritou.
- Zabezpečovanie garancie prenosovej šírky. Toto je bežne dosiahnuté nastavením určitého objemu prenosovej šírky (tzv. garantovaná prenosová šírka) s vysokou prioritou a prenos prevyšujúci garanciu s tou istou prioritou ako akýkoľvek iný prenos, ktorý konkuruje zvyšku prenosovej šírky bez nastavenia priorít.

Dobre nastavená regulácia dátového prenosu zvyčajne nepracuje na základe zoraďovania obrovského množstva údajov a následne zoraďovaných podľa priority prenášaných dát pred prenášanými dátami bez priority. Skôr zvyknú merať množstvo údajov zoradených podľa dôležitosti ako obmedzovanie nezoradeného prenosu dynamických závislostí tak, že nebude zasahovať do priepustnosti údajov s nastavenou prioritou.

4.3 Regulátor prenosu dát má nasledovné kľúčové vlastnosti:

„Pipe" princíp

Regulácia prenosu dát je zabezpečovaná na princípe zásobníkov, kde každý zásobník má niekoľko možností na nastavenie priorít, limitov a zoskupení. Jednotlivé zásobníky môžu byť prepojené rôznymi spôsobmi na skonštruovanie ovládania jednotiek, ktoré presahujú možnosti jedného zásobníka.

Spoločná integrácia s nastaveniami práv na Firewall

Každé pravidlo na Firewall-e môže byť priradené jednému alebo viacerým zásobníkom osobitne.

Nastavenie priorít prenosu a limitovanie prenosovej šírky

Každý zásobník obsahuje niekoľko úrovní priorít, každý z nich má vlastné obmedzenie prenosovej šírky, špecifikované v kilobitoch za sekundu a/alebo v paketoch za sekundu. Obmedzenia môžu byť takisto dané pre celý zásobník.

Zoskupovanie

Prenos dát cez zásobní môže byť automaticky zoskupený do užívateľov zásobníkov, kde každý užívateľ zásobníku môže byť nakonfigurovaný na takú istú šírku ako hlavný zásobník.

Prenos dát môže byť zoskupený podľa niekoľkých parametrov, napríklad zdroj alebo cieľová sieť, IP adresa alebo číslo portu.

Dynamické priraďovanie prenosovej šírky

Regulátor prenosu môže byť používaný na dynamické zmeny prenosovej šírky rôznych užívateľov zásobníkov, v prípade, že zásobník ako celok presiahne svoje limity.

To znamená, že poskytnutá šírka prenosu je prípadne obmedzená s ohľadom na vybranú skupinu zásobníkov.

Reťazce zásobníkov
Pokiaľ sú zásobníkom priradené pravidlá, najviac 8 zásobníkov môže byť prepojených do jedného reťazca. Toto umožňuje veľmi sofistikované filtrovanie a obmedzovanie.

Garancie prenosu dát

S vhodnou konfiguráciou zásobníka môže byť regulácia prenosu dát použitá na garantovanie prenosovej šírky (a tým pádom kvality) pre prenos cez firewall.

4.4 Spôsoby správy prenosovej šírky

Pravidlá podľa aplikácie

Regulátor prenosu môže identifikovať a kategorizovať konkrétne typy prenosu na sieti, obmedzujúc každú určitú kategóriu prenosu použitím nie viac než špecifikovaného objemu prenosovej šírky. Napríklad, hypoteticky je možné mať pravidlo, ktoré obmedzuje kompletný prenos FTP služby na nie viac 6 megabitov ua sekundu a ďalšie pravidlo, ktoré obmedzuje celkový prenos streaming audio na nie viac ako 3 Mbps, atď.

Regulátor prenosu môže kategorizovať prenos na základe makroskopických charakteristík, ako sieťový protokol (IP, IPX, Apple Talk, DECNet, atď), porty požívané aplikáciami (ako napríklad Kazaa zvyčajne beží na porte 1214) alebo na základe pripojení do bežne známych hostiteľov (ako centrálne hracie servery)m atď.

Prenos môže byť taktiež kategorizovaný podľa obsahu prietoku nezohľadňujúc makroskopické charakteristiky prietoku. Väčšina regulátorov prenosu môžu jednoducho identofikovať a automaticky kategorizovať prenos webových stránok, čo je pri vyžiadaní stránky založené na komunikácii medzi serverom webových stránok a webovým prehliadačom, bezohľadu na to, či server webových stránok beží na porte 80 (štandardné nastavenie) alebo na inom neštandardnom porte.

Pravidlá podľa užívateľa

Regulátory prenosu môžu nastaviť obmedzenia prenosu podľa užívateľa tak, aby zabezpečili prenos po sieti rovnomerne pre všetkých užívateľov. Napríklad, môžeme sa rozhodnúť použiť pravidlo podľa užívateľa, ktoré obmedzuje prenos od alebo ku každému užívateľovi na maximálne 256Kbps (poskytneme im príklad DSL služby). Ak je prenos obmedzovaný takýmto spôsobom, užívateľ má stále prístup ku všetkému ale prietoky sú obmedzené na danú úroveň radšej ako možnosť použitia celej prenosovej kapacity pripojenia na Internet.

Prenosové obmedzenia môžu byť pevné ako aj navyšovateľné. Ako by sme mohli čakať, pevný limit je určený strop, ktorý nemôže byť prekročený. Na druhej strane, navyšovateľné limity povoľujú taký prenos, ktorý môže prekročiť danú hraničnú hodnotu pokiaľ kapacita ostáva k dispozícii a nie je iná priorita na využitie tejto zvyšnej kapacity.

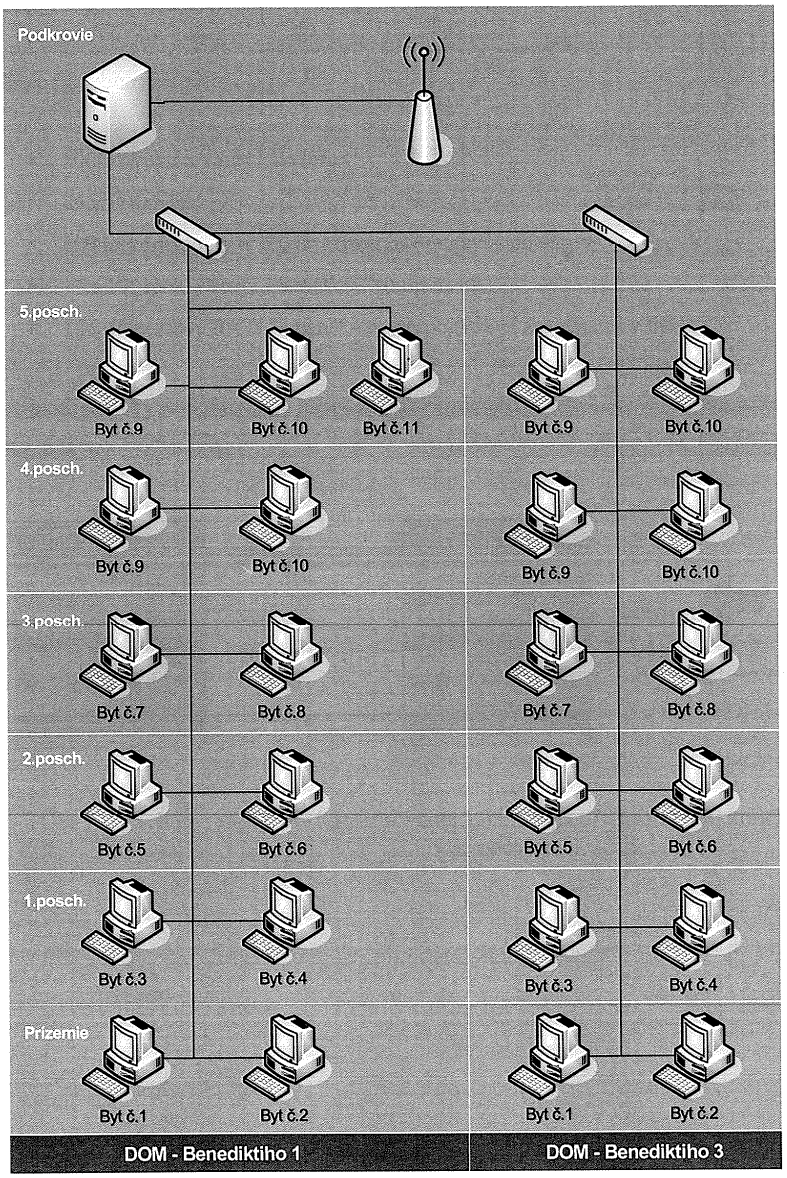
Správa priorít

Navyše k nastavovanie pevných alebo navyšovateľných prenosových obmedzení pri aplikačných alebo užívateľských základoch, zariadenia na reguláciu prenosu môžu byť tiež použité na definovanie vzájomnej dôležitosti rôznych typov prenosu. Napríklad na univerzitnej sieti, kde výučba a výskum sú najdôležitejšie, pasívne užívanie siete (ako počítačové hry alebo peer-to-peer aplikácie) môžu dostať prenosovú šírku iba v prípade, že aplikácia s vyššou prioritou ju nepotrebuje.

Niektoré úlohy na regulácie prenosu môžu byť riešené priamo na aktívnych sieťových prvkoch - router-och, takisto ako router môže byť použitý ako firewall s úlohou na filtrovanie paketov. Aj keď, špecializované regulátory prenosu, ako ktorékoľvek špecializované zariadenia, môžu byť optimalizované na špecifické a efektívne spracovávanie ich jedinečných úloh.

5 FYZICKÉ ZAPOJENIE SYSTÉMU

5.1 Schéma zapojenia siete domov Benediktiho 1 a 3



Obr 6. - Schéma logického zapojenia bytových prípojok

5.2 Hardwarové prvky použité v sieti

5.2.1 Zapojenie pasívnych prvkov siete

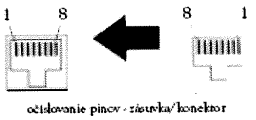
Na vybudovanie pasívnej siete a natiahnutie kabeláží

Bolo potrebné zakúpiť:

- 500m štruktérovanej kabeláže
- 25 ks zásuviek
- 100 ks konektorov a 100 ks kritiek na koncové konektory
- 1 ks 19" rack – skriňu na sieťové komponenty
- 1 ks 48 portový PATCH panet – na prepojenie so zasuvkami v bytoch

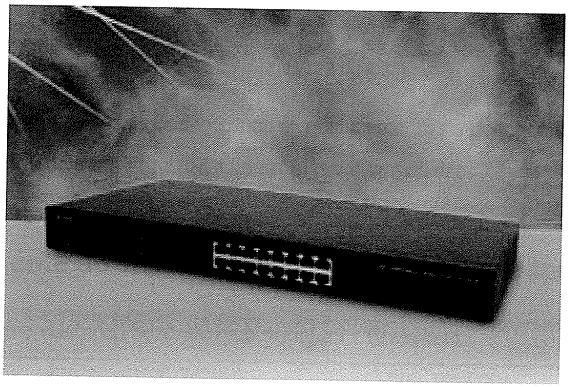
Jednotlivé linky štruktúrovanej kabeláže boli pripájané do konektorových koncoviek nasledovným spôsobom:

T568A			T568B	
Pin	Farba	Prenášaný signál	Pin	Farba
1	biela-zelená	TD+	1	biela-oranžová
2	zelená	TD-	2	oranžová
3	biela-oranžová	RD+	3	biela-zelená
4	modrá	-	4	modrá
5	biela-modrá	-	5	biela-modrá
6	oranžová	RD-	6	zelená
7	biela-hnedá	-	7	biela-hnedá
8	hnedá	-	8	hnedá



Obr 7. - Schéma zapojenia štruktúrovanej kabeláže podľa ktorej boli konektorované konsovky

5.2.2 Fast Ethernet přepínač NOVA SWITCH



Obr 8. - 16-portový prepínač PLANET FNSW-1601

Riešenie pre zaťažené siete typu client/server i peer-to-per. Veľmi prijateľné cenové podmienky a jednoduchý prechod medzi 10Mbps a 100Mbps sieťou. Prepínače NOVA Switch kombinujú 16 až 32 portov prepínaného Fast Ethernetu. Rovnako tak sú k dispozícii i modely s jedným až ôsmimi optickými portami Fast Ethernetu. Všetky porty prepínačov sú plne duplexné, Fast Ethernetové porty sú vybavené automatickou detekciou 10/100Mbps.

Technická špecifikácia použitého prepínača FNSW-1601 firmy PLANET

- **16 portov** 100 Base-TX, Switching, Full Duplex (100/200 Mbps), Auto-Negotiation (10/100 Mbps)
- 8k MAC adries, buffer 32kB na každý port
- 4k MAC adries, buffer 32kB na každý port
- 1 uplink port (MDI II) 100Base-TX
- LED diagnostika
- prevedenie rackmount (19") alebo inštalácia stôl (desktop)
- interný zdroj 220/50Hz

5.3 Konfigurácia počítača - router/firewall

Starší stolový počítač:

- Procesor Intel pentium

- 2x sieťový adaptér 100/10Mbit/s – s integrovanou funkciou ITR[9]
- 256 MB RAM
- 20 GB HDD

Táto lacná konfigurácia, je dostatočne výkonná na obslúženie 25 staníc v počítačovej sieti a vykonávať nad nimi ochranný firewall a kontrolu prenášaných dát

5.4 Pripojenie do siete internet cez mikrovlnné spojenie

Tab. 4. - Porovnanie vlastností bezdrátových pripojení v Bratislave

Bluetooth	2 Mb/s	2,4 GHz	slabý výkon, rýchlosť	cena, podpora zariadení, univerzálnosť
802.11	2 Mb/s	2,4 GHz	rýchlosť	stabilná prenosová rýchlosť
802.11a	54 Mb/s	5 GHz	nedostupné zariadenia, cena	nezarušené pásmo, rýchlosť, bezpeznosť
802.11b	11 Mb/s	2,4 GHz	preplnené pásmo, veľmi nízka bezpečnosť	cena, maximálna podpora v OS Linux
802.11b+	22 Mb/s	2,4 GHz	preplnené pásmo	vyššia bezpečnosť
802.11g	54 Mb/s	2,4 GHz	preplnené pásmo	rýchlosť

Prameň: HEČKO, M. *Wi–Fi už nie je len vízia. PC REVUE,* 2003, roč 11, č. 10, str. 24-25

Pripojenie nám sprostredkovala firma SWAN, a.s.

Ako sa nám potvrdilo z praxe, v Bratislave a hlavne v centre mesta je verejné pásmo 2,4GHz preplnené. Po pripojení bez drátového spoju sa nám z plánovaných 2 MBit/s podarilo dostať maximálne na 1,3 MBit/s.

Preto, sme museli prejsť na mikrovlnné spojenie tiež tej istej prenosovej šírky ale na pásme 5,6 MHz, ktoré je navyše bezpečnejšie. Vyjednali sme výhodné cenové podmienky, preto cena nebola výraznou zmenou voči plánu.

[9] Interrupt Throtling

Obr 9. - Access point na budove Stavebnej fakulty STU

5.5 Softvérové vybavenie

Kôli dodržaniu podmienky šetrenia nákladov bolo použité open source programové vybavenie na báze unixového operačného systému.

Open source

Ide o softvér s voľne dostupnými zdrojovými kódmi. Tento pojem však nedefinuje len dostupnosť zdrojových kódov softvéru, ale aj ďalšie aspekty vývoja. Ide o celkový prístup k vývoju softvéru a k otázkam komerčného charakteru spojených s predajom a distribúciou softvéru. Azda najznámejším open source je operačný systém Linux.

Základné body definujúce otvorený softvér:

- Dostupnosť zdrojového kódu: Zdrojový kód musí byť distribuovaný spolu s programom alebo musí byť definované, kde ho možno získať.
- Voľná redistribúcia: Softvér musí byť voľne distribuovateľný a za túto distribúciu nie je možné požadovať poplatky (výrazne) prevyšujúce náklady na túto distribúciu. Distribúciu nie je možné viazať na iné produkty.
- Voľná modifikácia: Softvér je možné modifikovať a odvádzať z neho nové produkty.
  · Žiadna diskriminácia: Použitie softvéru nesmie diskriminovať žiadnu osobu alebo skupinu osôb

5.5.1 Operačný systém Linux

Prečo bol vybratý Linux:

- ide o mimoriadne stabilný, kvalitný a výkonný operačný systém
- kôli prístupnosti zdrojových kódov prechádza neustálym a nezávislým bezpečnostným auditom verejnosti a aj gigantov typu IBM alebo SAP
- má výhodné licenčné podmienky a vďaka nim, môžete ušetriť desiatky až stovky tisíc korún

Firewall beží na operačnom systéme DebianGNU/Linux s jadrom verzie 3.1 obsluhuje služby NAT, DNS, trafic shaping, firewall

5.6 Regulácia toku dát

„Zoraďovaním zisťujeme, ktoré dáta sú poslané. Podstatné je zistenie, že môžeme regulovať iba rozhranie prenosu. Spôsob, akým pracuje Internet, nemá priamu kontrolu toho, čo užívatelia pošlú. Je to niečo podobné ako vaša poštová schránka na dome. Neexistuje spôsob, ako by ste mohli modifikovať množstvo pošty, ktorá vám má prísť, tak, aby ste nemuseli každého predom kontaktovať.

Aj keď, Internet je zväčša postavený na TCP/IP, ktorý má pár vlastností, ktoré nám môžu pomôcť. TCP/IP nedokáže zistiť kapacitu siete medzi dvoma hostiteľmi a preto jednoducho začne posielať dáta rýchlejšie a rýchlejšie (slow start) a keď sa začnú pakety strácať, pretože už nie je možnosť ich poslať, prenos sa spomalí.

Toto je ekvivalent k tomu, keď by ste napríklad neprečítali polovicu svojich e-mailov a dúfali, že ľudia vám ich prestanú posielať. S tým rozdielom, že to v Internete funguje.

Pokiaľ je router k dispozícii a je potreba chrániť sa pred konkrétnym užívateľom v rámci siete pred sťahovaním príliš veľkého objemu dát, musí sa regulovať vnútorný vstup na router-i, ktorý posiela dáta.

Takisto je nutné zabezpečiť kontrolu na strope prenosového pásma. Pokiaľ je použitá 100Mbit sieťový adaptér a je použitý router s 2Mbit, nesmú sa posielať viac dát, než je router schopný spracovať. V opačnom prípade bude router ten, ktorý kontroluje a reguluje šírku prenosovej linky. Je potrebné vytvoriť vlastný rad (hrdlo), aby bolo možné byť najpomalším článkom na prepojení lokálnej siete a pripojením do internetu."[10]

To, čo bolo povedané, je potrebné zistiť, že všetky prichádzajúce údaje sú regulované na sieťovom adaptéri eth0 a všetky odchádzajúce údaje sú regulované na eth1. Napríklad, keď je regulovaný prichádzajúci tok údajov, eth1 pustí tieto údaje na eth0 bez starostlivosti o regulovaní prenosu. Sú spôsoby, ako regulovať prichádzajúce údaje, ale odkedy sú kontrolované obe strany router-a, tak sme schopní regulovať iba výstupné údaje na každom sieťovom rozhraní.

My používame linux bridge medzi router-om a LAN. Bridge používa HTB[11] a qdisc[12] so siedmymi triedami prenosu údajov. Týchto 7 tried je nasledovných:

Trieda 10 - Neobmedzene (plný prístup)

Táto trieda je pre prenos údajov, ktorý nesmie byť pozdržaný pre prípad, že užívateľ sa začne sťažovať na pomalosť siete. Prenos údajov, ktorý spadá do tejto triedy, je nasledovný:

* ftp-data (20)
* ftp (21)
* ssh (22)
* telnet (23)
* Various Specific IPs
* VoIP

Trieda 20 – Mierna obmedzenosť (1mbit minimum, 1.5mbit maximum)

Táto trieda je pre prenos údajov, o ktorý užívateľ nemá prehľad a pre online prenos spojený s počítačovými sieťovými hrami. Prenos údajov, ktorý spadá do tejto triedy, je nasledovný:

* HTTP (80)

---

[10] Prameň: HUBERT, B. , *Linux Advanced Routing & Traffic Control HOWTO*, DocBook Edition, str. 24, 20.7.2002

[11] Heirarchical Token Bucket

[12] Queueing Discipline - riadiaca disciplína obsahujúca triedy

* pop3 (110)
* imap (143)
* ldap (389)
* https (443)
* port 522
* rtsp (554)
* imaps (993)
* Battlenet (4000)
* Instant Messenger (5190, 5191)
* Battlenet (6112)
* realmedia (7070)
* realmedia (7078)
* http games (8080)
* yahoo games (11999)

Trieda 30 - NNTP prenos (80kbit minimum, 200kbit maximum)

Trieda 40 - Nešpecifikovaný prenos (20kbit minimum, 1mbit maximum)

Táto trieda je pre prenos údajov, ktoré nie sú zaradené do žiadnej triedy. Táto trieda je prevažná časť prenosu všetkých údajov.

Trieda 50 - ICMP (128kbit maximum)

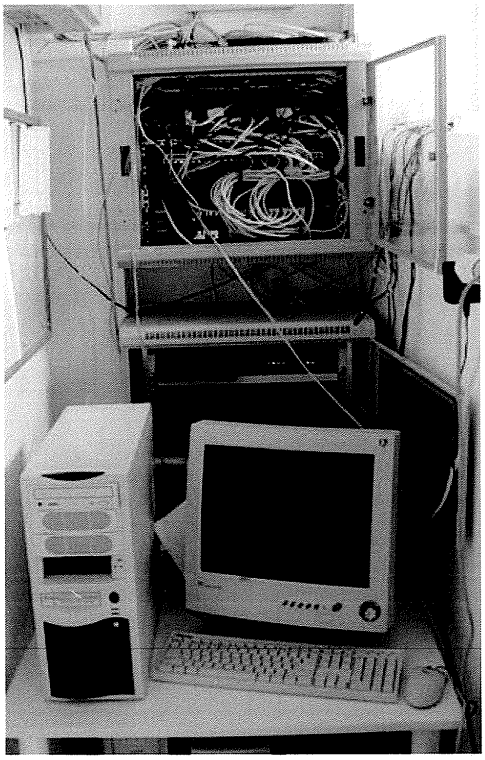
Trieda 60 – prenos údajov spojené so známymi vírusmi a červami (1kbit minimum, 5kbit maximum)

Trieda 70 - SJC IPs definovaná ako "Zneužívatelia siete" (1kbit minimum, 75kbit maximum)

Táto trieda je pre počítače, ktoré boli priradené do skupiny, ktorá zneužíva sieť. Na sieti beží program, ktorý určuje množstvo sieťových zdrojov v závislosti od jednotlivých užívateľov. Pokiaľ chovanie užívateľa na sieti nie je korektné voči ostatným užívateľom, program zaradí tohto užívateľa do tejto triedy. Zvyčajne sem zaraďuje užívateľov využívajúcich sieťový prenos na Peer-to-Peer aplikácie (napr. kazaa, limewire, morpheus, atď.)

5.7 Serverová miestnosť

Umiestnenie serverovej miestnosti je v podkroví čo naj blyžšie ku anténe MW spojenia.



Obr 10. - Serverová miestnosť

5.8 Konfiguračný skript na nastavenie regulácie toku dát

Skopíruje sa do adresára etc\init.d a nazveme si ho shaper,. Vytvoríme symbolický link etc/rc3.d a jeho názov je S17shaper. Vdaka tomuto nastaveniu sa po reštarte bude automaticky spúšťať shaper.

Podrobný výpis skriptu sa nachádza v prílohe 1.

5.9 Konfiguračný skript na nastavenie firewallu

Skopíruje sa do adresára etc\init.d a nazveme si ho firewall,. Vytvoríme symbolický link etc/rc3.d a jeho názov je S18firewall. Vďaka tomuto nastaveniu sa po reštarte bude automaticky spúšťať firewall.

Podrobný výpis skriptu sa nachádza v prílohe 2.

6 HODNOTENIE UŽÍVATEĽOV

p. Jurkovič, obyvateľ domu na Benediktiho ulici 3 v Bratislave

Prechodom na naš „domový internet" z Dial-up pripojenia sa tak markantne zvýšila rýchlosť načítania www stránok a e-mailovej pošty, že v súčasnosti počítač čaká na moje príkazy a nie ja ako užívateľ na načítanie údajov z internetu. Takisto je ďaleko pohodlnejšie pripájať sa kedykoľvek bez ohľadu na tarifné pásmo telefónnej linky a počítať čas pripojenia. To čo bolo hlavným cieľom, znížili sa mi poplatky za pripojenie na internet na paušálnych 500,- Sk. Hodnotím tento projekt ako pohodlné a z dlhodobého hľadiska finančne výhodné riešenie

p. Piatkovský, obyvateľ domu na Benediktiho ulici 1 v Bratislave

Pred rozhodnutím budovať centrálne internetové pripojenie som používal pripojenie cez ISDN linku. Rýchlosť pripojenia bola pre mňa v podstate dostačujúca, ale pripojenie cez 2Mbitové mikrovlnné spojenie napriek tomu, že sa oň delí 25 bytov predčilo moje očakávania. Bol to výborný nápad realizovať takéto pripojenie. Takisto úvodný poplatok (1600 Sk) bol v podstate symbolický vzhľadom na to, že sa tým pokryli všetky náklady a vytvoril sa malý rezervný fond na prípadné opravy.

ZÁVER

Zámerom mojej bakalárskej práce bolo počítačovou sieťou prepojiť a centrálne pripojiť 2 bytové domy do siete internet s čo možno najnižšími finančnými nákladmi.

V úvode tejto práce som sa zaoberal potrebami, ktoré viedli k tomuto riešeniu a definoval som problémy, ktoré majú byť vyriešené.

Vo svojej práci opisujem princípy a riešenia prepojenia domácností do počítačovej siete, bezpečného pripojenie do internetu, a rozdeľovania prenosového pásma.

Toto riešenie pokrýva narastajúce potreby ľudí mať zavedené bezpečné pripojenie na internet aj doma. Napriek prvotne nevhodne vybratému mikrovlnnému pripojeniu vo verejnom pásme 2,4Ghz a následnému prechodu na pásmo v 5,6 Ghz, považujem tento projekt za úspešne dokončený a spĺňa všetky požiadavky užívateľov. Som rád, že mám v kompetencii spravovať, kontrolovať a dolaďovať vytvorenú počítačovú sieť.

Mojou víziou je využiť existujúcu sieť a súčasné prepojenie domácností aj na prenos hlasových služieb a nahradiť tým pevne telefónne linky zavedené do domácností.

RESUMÉ

V dnešnej dobe zaznamenávame rýchly vzostup penetrácie prístupu na internet do domácností. Trendom je nielen pokles cien za poskytnuté služby na pripojenie do internetu, ale aj zvyšovanie rýchlosti a kvality pripojenia. S narastajúcimi nárokmi užívateľov a vývojom informačných technológií je nutné zvyšovať bezpečnosť počítačových staníc pripojených do internetu.

Súčasný trh ponúka rôzne technologické a marketingové produktu a kompletné balíky služieb spojené s internetovým pripojením. Tento priojekt vznikol na základe snahy znížiť náklady spojené s dostupnosťou na internet v domácnostiach.

Táto bakalárska práca sa zaoberá riešením pripojenia pre 25 bytov rozdelených v dvoch susediacich domoch. Riešenie je rozdelené do návrhu a realizácie lokálnej siete spájajúcej všetky spomínané byty, výber vhodného a cenovo najdostupnejšieho pripojenia na internet a inštaláciu a konfiguráciu ochranného Firewallu umiestneného medzi lokálnou sieťou a internetom. Taktiež výber hardwarového softwarového vybavenia pre Firewall, ktoré majú spĺňať požiadavku na čo najnižšie náklady.

Výber mikrovlnného pripojenia v 5,6Ghz pásme sa ukázalo ako vysokorýchlostné, stabilné riešenie, ktoré na jednu domácnosť v porovnateľnej kvalite znížilo náklady na internetové pripojenie približne o 60%. Neskôr je možné rozšíriť takto vybudovanú sieť na komplexné hlasové aj dátové služby a nahradiť tým pevné linky.

In present we are witneses of significant increase of penetration of acces on Internet into households. A trend is not just drop in prices for provided services concernig connection to Internet, but also enhancing speed and a quality of connection. Together with increasing requirements of users and with development and progress of information technology it is necessary to intensify and increase the safety of PC stations connected to Internet.

Current market offer various technologic and marketing product and complex paskages of services associated with internet access and connection. This project originated on the base of effort to decerase expenses associated with accesibility on Internet in households.

This thesis is concerned with a solvation os connection for 25 flats devided in 2 neighbouring buildings. The solution is devided into the proposal and realisation of local network connecting mentioned buildings, a choise of suiatble and the most price reasonable connection to Internet and installation and configuration of protective Firewall places between local network and Internet. It also includes a choice of hardware and software equipment for Firewall, which should fill in the condition of the lowest cost.

A choice of microwave connection within 5,6Ghz zone has been proved to be high-speeded, stable solution, which has decerased costs on Internet connection approximately about 60% Within comparable quality per 1 household. Lates it is possible to upgrade such a built network to complex voice and data services and thereby phone lines can be replaced.

ZOZNAM POUŽITEJ LITERATÚRY A ĎALŠÍCH PRAMEŇOV

HEČKO, M. *Wi−Fi už nie je len vízia. PC REVUE,* 2003, roč 11, č. 10, str. 24-25

PETŘÍČEK, M. *Stavíme firewall (3).* [on-line].
Dostupný na WWW: http://www.root.cz/clanky/stavime-firewall-3/ [cit. 10. júna 2005]

PETERKA, J. *TCP/IP a vzájemné propojování sítí* [on-line]
Dostupný na WWW: http://www.earchiv.cz//a92/a232c110.php3 [cit. 10. júna 2005]

HORÁK, J. *Bezpečnost malých počítačových sítí*, 1. vyd., GRADA Publishing, 2003, 200 s, ISBN: 80-247-0663-6

DOSTÁLEK, L. A KOLEKTIV. Velký průvodce protokoly TCP/IP: bezpečnost, 1. vyd., Computer Press 2003, 592 s, ISBN: 80-7226-849-X

JIROVSKÝ, V. *Vademecum správce sítě*, 1. vyd., GRADA Publishing, 2001, 428 s, ISBN: 80-7169-745-1

STREBE, M, PERKINS, C. *Firewally a proxy-servery: Praktický průvodce*, 1. vyd., Computer Press, 2003, 472 s, ISBN: 80-722-6983-6

HONTANÓN J.H. *Linux - praktická bezpečnost*, 1. vyd., GRADA Publishing, 2003, 440 s, ISBN: 80-247-0652-0

Zoznam príloh

Príloha 1 – konfiguračný skript na nastavenie regulácie prenosu dát

Príloha 2 – konfiguračný skript na nastavenie firewallu

Príloha 1 – konfiguračný skript na nastavenie regulácie prenosu dát

```
export LC_ALL=C

### Command locations
TC=/sbin/tc
IP=/sbin/ip
MP=/sbin/modprobe

### Default filter priorities (must be different)
PRIO_RULE_DEFAULT=${PRIO_RULE:-100}
PRIO_MARK_DEFAULT=${PRIO_MARK:-200}
PRIO_REALM_DEFAULT=${PRIO_REALM:-300}

### Default CBQ_PATH &amp;amp; CBQ_CACHE settings
CBQ_PATH=${CBQ_PATH:-/etc/sysconfig/cbq}
CBQ_CACHE=${CBQ_CACHE:-/var/cache/cbq.init}

### Uncomment to enable logfile for debugging
#CBQ_DEBUG="/var/run/cbq-$1"

### Modules to probe for. Uncomment the last CBQ_PROBE
### line if you have QoS support compiled into kernel
CBQ_PROBE="sch_cbq sch_tbf sch_sfq sch_prio"
CBQ_PROBE="$CBQ_PROBE cls_fw cls_u32 cls_route"
#CBQ_PROBE=""

### Keywords required for qdisc &amp;amp; class configuration
CBQ_WORDS="DEVICE|RATE|WEIGHT|PRIO|PARENT|LEAF|BOUNDED|ISOLATED"
CBQ_WORDS="$CBQ_WORDS|PRIO_MARK|PRIO_RULE|PRIO_REALM|BUFFER"
CBQ_WORDS="$CBQ_WORDS|LIMIT|PEAK|MTU|QUANTUM|PERTURB"


################################################################################
############################## SUPPORT FUNCTIONS ###############################
################################################################################

### Get list of network devices
cbq_device_list () {
       ip link show| sed -n "/^[0-9]/ \
              { s/^[0-9]\+: \([a-z0-9._]\+\)[:@].*/\1/; p; }"
} # cbq_device_list


### Remove root class from device $1
cbq_device_off () {
       tc qdisc del dev $1 root 2&gt; /dev/null
} # cbq_device_off


### Remove CBQ from all devices
cbq_off () {
       for dev in `cbq_device_list`; do
              cbq_device_off $dev
       done
} # cbq_off


### Prefixed message
cbq_message () {
       echo -e "**CBQ: $@"
} # cbq_message

### Failure message
cbq_failure () {
       cbq_message "$@"
       exit 1
```

```
} # cbq_failure

### Failure w/ cbq-off
cbq_fail_off () {
       cbq_message "$@"
       cbq_off
       exit 1
} # cbq_fail_off


### Convert time to absolute value
cbq_time2abs () {
       local min=${1##*:}; min=${min##0}
       local hrs=${1%%:*}; hrs=${hrs##0}
       echo $[hrs*60 + min]
} # cbq_time2abs


### Display CBQ setup
cbq_show () {
       for dev in `cbq_device_list`; do
              [ `tc qdisc show dev $dev| wc -l` -eq 0 ] &amp;amp;&amp;amp; continue
              echo -e "### $dev: queueing disciplines\n"
              tc $1 qdisc show dev $dev; echo

              [ `tc class show dev $dev| wc -l` -eq 0 ] &amp;amp;&amp;amp; continue
              echo -e "### $dev: traffic classes\n"
              tc $1 class show dev $dev; echo

              [ `tc filter show dev $dev| wc -l` -eq 0 ] &amp;amp;&amp;amp; continue
              echo -e "### $dev: filtering rules\n"
              tc $1 filter show dev $dev; echo
       done
} # cbq_show


### Check configuration and load DEVICES, DEVFIELDS and CLASSLIST from $1
cbq_init () {
       ### Get a list of configured classes
       CLASSLIST=`find $1 \( -type f -or -type l \) -name 'cbq-*' \
              -not -name '*~' -maxdepth 1 -printf "%f\n"| sort`
       [ -z "$CLASSLIST" ] &amp;amp;&amp;amp;
              cbq_failure "no configuration files found in $1!"

       ### Gather all DEVICE fields from $1/cbq-*
       DEVFIELDS=`find $1 \( -type f -or -type l \) -name 'cbq-*' \
                -not -name '*~' -maxdepth 1| xargs sed -n 's/#.*//; \
                s/[[:space:]]//g; /^DEVICE=[^,]*,[^,]*\(,[^,]*\)\?/ \
                { s/.*=//; p; }'| sort -u`
       [ -z "$DEVFIELDS" ] &amp;amp;&amp;amp;
              cbq_failure "no DEVICE field found in $1/cbq-*!"

       ### Check for different DEVICE fields for the same device
       DEVICES=`echo "$DEVFIELDS"| sed 's/,.*//'| sort -u`
       [ `echo "$DEVICES"| wc -l` -ne `echo "$DEVFIELDS"| wc -l` ] &amp;amp;&amp;amp;
              cbq_failure "different DEVICE fields for single device!\n$DEVFIELDS"
} # cbq_init


### Load class configuration from $1/$2
cbq_load_class () {
       CLASS=`echo $2| sed 's/^cbq-0*//; s/^\([0-9a-fA-F]\+\).*/\1/'`
       CFILE=`sed -n 's/#.*//; s/[[:space:]]//g;
/^[[:alnum:]_]\+=[[:alnum:].,:;/*@-_]\+$/ p' $1/$2`

       ### Check class number
       IDVAL=`/usr/bin/printf "%d" 0x$CLASS 2&gt; /dev/null`
       [ $? -ne 0 -o $IDVAL -lt 2 -o $IDVAL -gt 65535 ] &amp;amp;&amp;amp;
```

```
                cbq_fail_off "class ID of $2 must be in range &lt;0002-FFFF&gt;!"

        ### Set defaults &amp; load class
        RATE=""; WEIGHT=""; PARENT=""; PRIO=5
        LEAF=tbf; BOUNDED=yes; ISOLATED=no
        BUFFER=10Kb/8; LIMIT=15Kb; MTU=1500
        PEAK=""; PERTURB=10; QUANTUM=""

        PRIO_RULE=$PRIO_RULE_DEFAULT
        PRIO_MARK=$PRIO_MARK_DEFAULT
        PRIO_REALM=$PRIO_REALM_DEFAULT

        eval `echo "$CFILE"| grep -E "^($CBQ_WORDS)="`

        ### Require RATE/WEIGHT
        [ -z "$RATE" -o -z "$WEIGHT" ] &amp;&amp;
                cbq_fail_off "missing RATE or WEIGHT in $2!"

        ### Class device
        DEVICE=${DEVICE%%,*}
        [ -z "$DEVICE" ] &amp;&amp; cbq_fail_off "missing DEVICE field in $2!"

        BANDWIDTH=`echo "$DEVFIELDS"| sed -n "/^$DEVICE,/ \
                { s/[^,]*,\([^,]*\).*/\1/; p; q; }"`

        ### Convert to "tc" options
        PEAK=${PEAK:+peakrate $PEAK}
        PERTURB=${PERTURB:+perturb $PERTURB}
        QUANTUM=${QUANTUM:+quantum $QUANTUM}

        [ "$BOUNDED" = "no" ] &amp;&amp; BOUNDED="" || BOUNDED="bounded"
        [ "$ISOLATED" = "yes" ] &amp;&amp; ISOLATED="isolated" || ISOLATED=""
} # cbq_load_class


################################################################################
##################################### INIT #####################################
################################################################################

### Check for presence of ip-route2 in usual place
[ -x $TC -a -x $IP ] ||
        cbq_failure "ip-route2 utilities not installed or executable!"


### ip/tc wrappers
if [ "$1" = "compile" ]; then
        ### no module probing
        CBQ_PROBE=""

        ip () {
                $IP "$@"
        } # ip

        ### echo-only version of "tc" command
        tc () {
                echo "$TC $@"
        } # tc

elif [ -n "$CBQ_DEBUG" ]; then
        echo -e "# `date`" &gt; $CBQ_DEBUG

        ### Logging version of "ip" command
        ip () {
                echo -e "\n# ip $@" &gt;&gt; $CBQ_DEBUG
                $IP "$@" 2&gt;&amp;1 | tee -a $CBQ_DEBUG
        } # ip

        ### Logging version of "tc" command
```

```
        tc () {
                echo -e "\n# tc $@" &amp;gt;&amp;gt; $CBQ_DEBUG
                $TC "$@" 2&amp;gt;&amp;amp;1 | tee -a $CBQ_DEBUG
        } # tc
else
        ### Default wrappers

        ip () {
                $IP "$@"
        } # ip

        tc () {
                $TC "$@"
        } # tc
fi # ip/tc wrappers


case "$1" in

################################################################################
############################### START/COMPILE ##################################
################################################################################

start|compile)

### Probe QoS modules (start only)
for module in $CBQ_PROBE; do
        $MP $module || cbq_failure "failed to load module $module"
done

### If we are in compile/nocache/logging mode, don't bother with cache
if [ "$1" != "compile" -a "$2" != "nocache" -a -z "$CBQ_DEBUG" ]; then
        VALID=1

        ### validate the cache
        [ "$2" = "invalidate" -o ! -f $CBQ_CACHE ] &amp;amp;&amp;amp; VALID=0
        if [ $VALID -eq 1 ]; then
                [ `find $CBQ_PATH -maxdepth 1 -newer $CBQ_CACHE| \
                  wc -l` -gt 0 ] &amp;amp;&amp;amp; VALID=0
        fi

        ### compile the config if the cache is invalid
        if [ $VALID -ne 1 ]; then
                $0 compile &amp;gt; $CBQ_CACHE ||
                        cbq_fail_off "failed to compile CBQ configuration!"
        fi

        ### run the cached commands
        exec /bin/sh $CBQ_CACHE 2&amp;gt; /dev/null
fi

### Load DEVICES, DEVFIELDS and CLASSLIST
cbq_init $CBQ_PATH


### Setup root qdisc on all configured devices
for dev in $DEVICES; do
        ### Retrieve device bandwidth and, optionally, weight
        DEVTEMP=`echo "$DEVFIELDS"| sed -n "/^$dev,/ { s/$dev,//; p; q; }"`
        DEVBWDT=${DEVTEMP%%,*};    DEVWGHT=${DEVTEMP##*,}
        [ "$DEVBWDT" = "$DEVWGHT" ] &amp;amp;&amp;amp; DEVWGHT=""

        ### Device bandwidth is required
        if [ -z "$DEVBWDT" ]; then
                cbq_message "could not determine bandwidth for device $dev!"
                cbq_failure "please set up the DEVICE fields properly!"
        fi
```

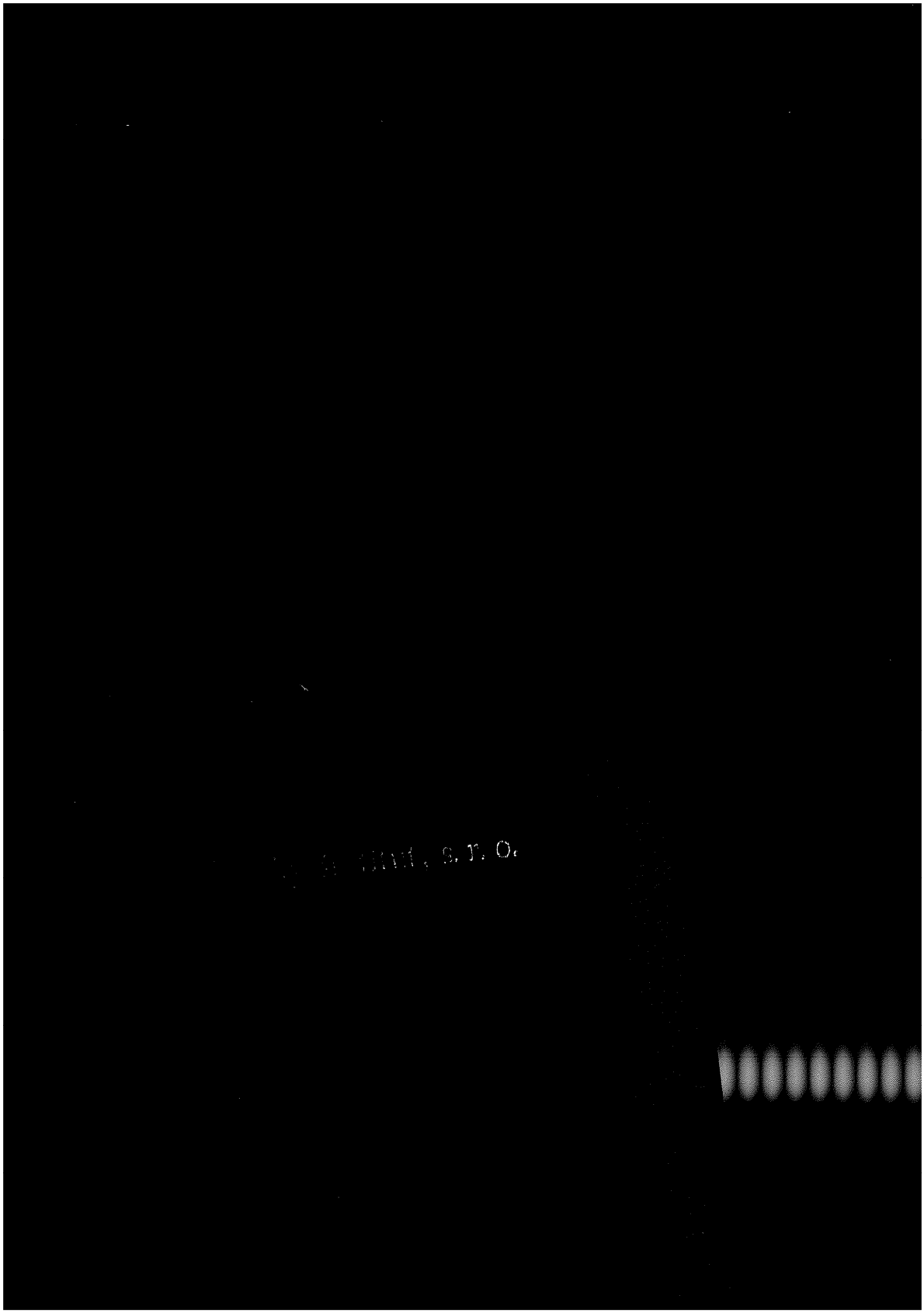
```
        ### Check if the device is there
        ip link show $dev &amp;&gt; /dev/null ||
                cbq_fail_off "device $dev not found!"

        ### Remove old root qdisc from device
        cbq_device_off $dev


        ### Setup root qdisc + class for device
        tc qdisc add dev $dev root handle 1 cbq \
        bandwidth $DEVBWDT avpkt 1000 cell 8

        ### Set weight of the root class if set
        [ -n "$DEVWGHT" ] &amp;&amp;
                tc class change dev $dev root cbq weight $DEVWGHT allot 1514

        [ "$1" = "compile" ] &amp;&amp; echo
done # dev


### Setup traffic classes
for classfile in $CLASSLIST; do
        cbq_load_class $CBQ_PATH $classfile

        ### Create the class
        tc class add dev $DEVICE parent 1:$PARENT classid 1:$CLASS cbq \
        bandwidth $BANDWIDTH rate $RATE weight $WEIGHT prio $PRIO \
        allot 1514 cell 8 maxburst 20 avpkt 1000 $BOUNDED $ISOLATED ||
                cbq_fail_off "failed to add class $CLASS with parent $PARENT on
$DEVICE!"

        ### Create leaf qdisc if set
        if [ "$LEAF" = "tbf" ]; then
                tc qdisc add dev $DEVICE parent 1:$CLASS handle $CLASS tbf \
                rate $RATE buffer $BUFFER limit $LIMIT mtu $MTU $PEAK
        elif [ "$LEAF" = "sfq" ]; then
                tc qdisc add dev $DEVICE parent 1:$CLASS handle $CLASS sfq \
                $PERTURB $QUANTUM
        fi


        ### Create fw filter for MARK fields
        for mark in `echo "$CFILE"| sed -n '/^MARK/ { s/.*=//; p; }'`; do
                ### Attach fw filter to root class
                tc filter add dev $DEVICE parent 1:0 protocol ip \
                prio $PRIO_MARK handle $mark fw classid 1:$CLASS
        done ### mark

        ### Create route filter for REALM fields
        for realm in `echo "$CFILE"| sed -n '/^REALM/ { s/.*=//; p; }'`; do
                ### Split realm into source &amp; destination realms
                SREALM=${realm%%,*}; DREALM=${realm##*,}
                [ "$SREALM" = "$DREALM" ] &amp;&amp; SREALM=""

                ### Convert asterisks to empty strings
                SREALM=${SREALM#\*}; DREALM=${DREALM#\*}

                ### Attach route filter to the root class
                tc filter add dev $DEVICE parent 1:0 protocol ip \
                prio $PRIO_REALM route ${SREALM:+from $SREALM} \
                ${DREALM:+to $DREALM} classid 1:$CLASS
        done ### realm

        ### Create u32 filter for RULE fields
        for rule in `echo "$CFILE"| sed -n '/^RULE/ { s/.*=//; p; }'`; do
                ### Split rule into source &amp; destination
                SRC=${rule%%,*}; DST=${rule##*,}
                [ "$SRC" = "$rule" ] &amp;&amp; SRC=""
```


S

```
        ### Replace leaf qdisc (if any)
        if [ "$LEAF" = "tbf" ]; then
                tc qdisc replace dev $DEVICE handle $CLASS tbf \
                rate $NEW_RATE buffer $BUFFER limit $LIMIT mtu $MTU $NEW_PEAK
        fi

        cbq_message "$TIME_NOW: class $CLASS on $DEVICE changed rate ($RATE_NOW -
&gt; $NEW_RATE)"
done ### class file
;;


################################################################################
################################## THE REST ####################################
################################################################################

stop)
        cbq_off
        ;;

list)
        cbq_show
        ;;

stats)
        cbq_show -s
        ;;

restart)
        shift
        $0 stop
        $0 start "$@"
        ;;

*)
        echo "Usage: `basename $0`
{start|compile|stop|restart|timecheck|list|stats}"
esac
```

Príloha 2 – konfiguračný skript na nastavenie firewallu

```
#!/bin/bash

#vonkajsia ipcka
OUTIP=
#vnutorna ipcka
INIP=192.168.0.1
INNET=192.168.0.0/24
OUTETH=eth0
INETH=eth1
IPTABLES=/sbin/iptables

# zrusenie starych
$IPTABLES -F INPUT
$IPTABLES -F FORWARD
$IPTABLES -t nat -F PREROUTING
$IPTABLES -t nat -F POSTROUTING

# defaultne nastavenie na DROP
$IPTABLES -P INPUT DROP
$IPTABLES -P FORWARD DROP

# tieto povolime
$IPTABLES -A INPUT -i lo -j ACCEPT
$IPTABLES -A INPUT -i $INETH -j ACCEPT

# ostatne ako syn pakety povolene
$IPTABLES -A INPUT -p tcp \! --syn -j ACCEPT
$IPTABLES -A INPUT -m state -p tcp --state ESTABLISHED,RELATED -j ACCEPT
$IPTABLES -A INPUT -p icmp -j ACCEPT

# dns
$IPTABLES -A INPUT -p udp --sport 53 -j ACCEPT
$IPTABLES -A INPUT -p udp --dport 53 -j ACCEPT

# sluzby na tcp podla destination portu
$IPTABLES -A INPUT -p tcp --dport 25 -j ACCEPT
$IPTABLES -A INPUT -p tcp --dport 110 -j ACCEPT
$IPTABLES -A INPUT -p tcp --dport 80 -j ACCEPT
$IPTABLES -A INPUT -p tcp --dport 443 -j ACCEPT
$IPTABLES -A INPUT -p tcp --dport 22 -j ACCEPT


#namapovanie konkretneho portu uzivatelovy
#vonkajsi port 5000
#vnutorny 3389
#ipadresa uzivatela 192.168.0.7
$IPTABLES -t nat -A PREROUTING -i $OUTETH -p tcp --dport 5000 -j DNAT --to
192.168.0.7:3389


# ostatne tcp SYN budeme rejectovat
$IPTABLES -A INPUT -p tcp --syn -j REJECT

# maskarada (NAT)
$IPTABLES -A FORWARD -i $INETH -j ACCEPT
$IPTABLES -A FORWARD -d $INNET -j ACCEPT
$IPTABLES -A POSTROUTING -t nat -s $INNET -o $OUTETH -j MASQUERADE

echo "1" > /proc/sys/net/ipv4/ip_forward
echo "0" > /proc/sys/net/ipv4/icmp_echo_ignore_all


#echo "Teraz daj /etc/init.d/iptables save active"
```



```
            ### Split destination into address, port &amp;amp; mask fields
            DADDR=${DST%%:*}; DTEMP=${DST##*:}
            [ "$DADDR" = "$DST" ] &amp;amp;&amp;amp; DTEMP=""

            DPORT=${DTEMP%%/*}; DMASK=${DTEMP##*/}
            [ "$DPORT" = "$DTEMP" ] &amp;amp;&amp;amp; DMASK="0xffff"


            ### Split up source (if specified)
            SADDR=""; SPORT=""
            if [ -n "$SRC" ]; then
                  SADDR=${SRC%%:*}; STEMP=${SRC##*:}
                  [ "$SADDR" = "$SRC" ] &amp;amp;&amp;amp; STEMP=""

                  SPORT=${STEMP%%/*}; SMASK=${STEMP##*/}
                  [ "$SPORT" = "$STEMP" ] &amp;amp;&amp;amp; SMASK="0xffff"
            fi


            ### Convert asterisks to empty strings
            SADDR=${SADDR#\*}; DADDR=${DADDR#\*}

            ### Compose u32 filter rules
            u32_s="${SPORT:+match ip sport $SPORT $SMASK}"
            u32_s="${SADDR:+match ip src $SADDR} $u32_s"
            u32_d="${DPORT:+match ip dport $DPORT $DMASK}"
            u32_d="${DADDR:+match ip dst $DADDR} $u32_d"

            ### Uncomment the following if you want to see parsed rules
            #echo "$rule: $u32_s $u32_d"

            ### Attach u32 filter to the appropriate class
            tc filter add dev $DEVICE parent 1:0 protocol ip \
            prio $PRIO_RULE u32 $u32_s $u32_d classid 1:$CLASS
      done ### rule

      [ "$1" = "compile" ] &amp;amp;&amp;amp; echo
done ### classfile
;;


################################################################################
################################## TIME CHECK ##################################
################################################################################

timecheck)

### Get time + weekday
TIME_TMP=`date +%w/%k:%M`
TIME_DOW=${TIME_TMP%%/*}
TIME_NOW=${TIME_TMP##*/}

### Load DEVICES, DEVFIELDS and CLASSLIST
cbq_init $CBQ_PATH

### Run through all classes
for classfile in $CLASSLIST; do
      ### Gather all TIME rules from class config
      TIMESET=`sed -n 's/#.*//; s/[[:space:]]//g; /^TIME/ { s/.*=//; p; }' \
            $CBQ_PATH/$classfile`
      [ -z "$TIMESET" ] &amp;amp;&amp;amp; continue

      MATCH=0; CHANGE=0
      for timerule in $TIMESET; do
            TIME_ABS=`cbq_time2abs $TIME_NOW`
```

```
        ### Split TIME rule to pieces
        TIMESPEC=${timerule%%;*}; PARAMS=${timerule##*;}
        WEEKDAYS=${TIMESPEC%%/*}; INTERVAL=${TIMESPEC##*/}
        BEG_TIME=${INTERVAL%%-*}; END_TIME=${INTERVAL##*-}

        ### Check the day-of-week (if present)
        [ "$WEEKDAYS" != "$INTERVAL" -a \
          -n "${WEEKDAYS##*$TIME_DOW*}" ] &amp;&amp; continue

        ### Compute interval boundaries
        BEG_ABS=`cbq_time2abs $BEG_TIME`
        END_ABS=`cbq_time2abs $END_TIME`

        ### Midnight wrap fixup
        if [ $BEG_ABS -gt $END_ABS ]; then
              [ $TIME_ABS -le $END_ABS ] &amp;&amp;
                    TIME_ABS=$[TIME_ABS + 24*60]

              END_ABS=$[END_ABS + 24*60]
        fi

        ### If the time matches, remember params and set MATCH flag
        if [ $TIME_ABS -ge $BEG_ABS -a $TIME_ABS -lt $END_ABS ]; then
              TMP_RATE=${PARAMS%%/*}; PARAMS=${PARAMS#*/}
              TMP_WGHT=${PARAMS%%/*}; TMP_PEAK=${PARAMS##*/}

              [ "$TMP_PEAK" = "$TMP_WGHT" ] &amp;&amp; TMP_PEAK=""
              TMP_PEAK=${TMP_PEAK:+peakrate $TMP_PEAK}

              MATCH=1
        fi
done ### timerule


cbq_load_class $CBQ_PATH $classfile

### Get current RATE of CBQ class
RATE_NOW=`tc class show dev $DEVICE| sed -n \
      "/cbq 1:$CLASS / { s/.*rate //; s/ .*//; p; q; }"`
[ -z "$RATE_NOW" ] &amp;&amp; continue

### Time interval matched
if [ $MATCH -ne 0 ]; then

      ### Check if there is any change in class RATE
      if [ "$RATE_NOW" != "$TMP_RATE" ]; then
            NEW_RATE="$TMP_RATE"
            NEW_WGHT="$TMP_WGHT"
            NEW_PEAK="$TMP_PEAK"
            CHANGE=1
      fi

### Match not found, reset to default RATE if necessary
elif [ "$RATE_NOW" != "$RATE" ]; then
      NEW_WGHT="$WEIGHT"
      NEW_RATE="$RATE"
      NEW_PEAK="$PEAK"
      CHANGE=1
fi

### If there are no changes, go for next class
[ $CHANGE -eq 0 ] &amp;&amp; continue

### Replace CBQ class
tc class replace dev $DEVICE classid 1:$CLASS cbq \
bandwidth $BANDWIDTH rate $NEW_RATE weight $NEW_WGHT prio $PRIO \
allot 1514 cell 8 maxburst 20 avpkt 1000 $BOUNDED $ISOLATED
```