TECHNICKÁ UNIVERZITA LIBEREC

PEDAGOGICKÁ FAKULTA

Katedra: Informatiky

Kombinace: Fyzika – Informatika

Učebnice IPX

výukový text pro kroužky informatiky na SŠ

Diplomová práce: 98-PF-KIN-02

Autor:

Vladimír Vais podpis: Vladimír Vais

Adresa:

nám. Republiky 998/9

277 11 Neratovice

Vedoucí práce: RNDr. Pavel Satrapa

Počet:

stran	obrázků	tabulek	příloh
112	4	5	2

V Liberci, dne 23. května 1998

Technická univerzita v Liberci
PEDAGOGICKÁ FAKULTA
461 17 LIBEREC 1, Hálkova 6 Telefon: 329 Telefax: 21301

Katedra: informatiky

ZADÁNÍ DIPLOMOVÉ PRÁCE

(závěrečného projektu)

diplomant: Vladimír V A I S

adresa: nám.Republiky 998/9, Neratovice, 277 11

obor: informatika

Název: Výukový text IPX pro kroužky informatiky na středních školách

Vedoucí práce: RNDr. Pavel Satrapa

Termín odevzdání: 30.5. ~~1997~~ 1998

Pozn. Podmínky pro zadání práce jsou k nahlédnutí na katedrách. Katedry rovněž specifikují zadání: východiska, cíle, předpoklady, metody zpracování, základní literaturu (zpravidla na rub tohoto formuláře). Zásady pro zpracování DP jsou k dispozici ve dvou verzích (stručné, resp. metodické pokyny) v UK TUL, na katedrách a na Děkanátě Pedagogické fakulty.

V Liberci dne 6.11. 1996

prof.Ing. Jan Ehleman, CSc.
vedoucí katedry

doc.RNDr. Jaroslav Vild
děkan

Převzal (diplomant):
Datum: 25.11.96 Podpis: Vladimír Vais

Prohlášení a poděkování

Prohlášení o původnosti práce:

Prohlašuji, že jsem diplomovou práci vypracoval samostatně a že jsem uvedl veškerou použitou literaturu.

Liberec, 23. května 1998 Podpis: .....Vais.....

Prohlášení o využívání výsledků DP:

Jsem si vědom těchto skutečností:

- diplomová práce je majetkem školy
- s diplomovou prací nelze bez svolení školy disponovat
- diplomová práce může být zapůjčena či objednána (kopie) za účelem využití jejího obsahu

Beru na vědomí, že po pěti letech si mohu diplomovou práci vyžádat v Univerzitní knihovně Technické univerzity v Liberci, kde bude uložena.

Jméno a příjmení: Vladimír Vais

Adresa: nám. Republiky 998/9, 277 11 Neratovice

Podpis: .....Vais.....

Poděkování:

Děkuji všem, kteří se zasloužili o zdárné vypracování této diplomové práce. Děkuji RNDr. Pavlu Satrapovi za vhodný námět, konzultace a vedení diplomové práce. Děkuji RNDr. Petru Kolářovi a programátorům z Internetu za užitečné jednotky pro Borland Pascal. Děkuji svému otci za vytištění práce. Všichni tito lidé napomohli zdárnému dokončení diplomové práce, za což jim ještě jednou patří můj co nejvřelejší dík.

Anotace

Učebnice IPX – výukový text pro kroužky informatiky na SŠ

Diplomová práce se zabývá formou učebnice pro střední školu problematikou protokolu IPX. Cílem práce je vytvořit užitečnou pomůcku, která tuto problematiku srozumitelně a čtivě vysvětlí studentovi střední školy.

Annotation

IPX Schoolbok – Teaching text for Information Science Circles at a High School

Diploma thesis is engaged in studying the IPX protocol at a high school. The aim of the thesis is creation of useful aid, what understandable and readable explains the problem to high school student.

Анотациа

Учебник IPX

Дипломная работа занимается формой учебника для средней школы проблемой протоколя IPX. Цель работы это полезное пособие, какое обьясняет эту проблему понятно и разборчибно ученику из средней школы.

Technické informace

Editor: QEdit Advanced Shareware

Sazba: LaTeX 2_ε

Grafika: Corel Draw! 3.0, BM2FONT 3.0

Ukázkové programy: Borland Pascal 7.0, Lister 1.0

Operační systém: Microsoft Windows 95, RedHat Linux 5.0 + DosEmu

Počítače:

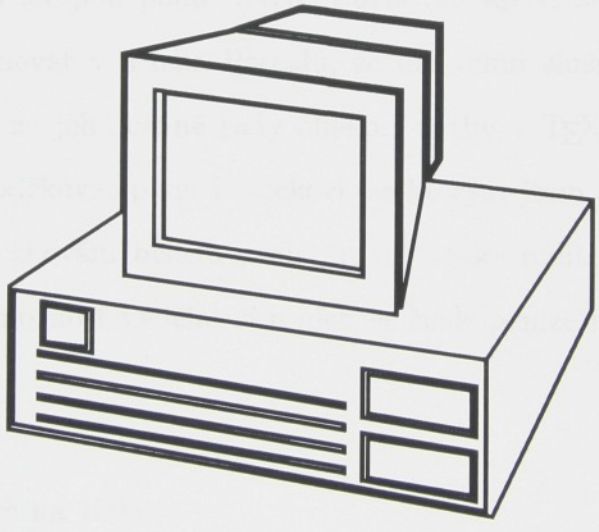
- AMD K5/90 MHz, 32 MB RAM, 3,1 GB HDD Western Digital, monitor 15" ADI MicroScan 4V, gr. karta S3 Trio 64V+ se 2 MB RAM, sériová myš Genius, paralelní kabel LapLink na portu ECP
- Intel Pentium MMX/250 MHz, 64 MB RAM, 3,1 GB HDD Seagate, monitor 15" AOC Spectrum 5 VIr, gr. karta ATI Mach64 se 2 MB RAM, sériová myš Genius, paralní kabel LapLink na portu ECP

Tisk: HP LaserJet 4L – 300 dpi

Licence na Corel Draw! 3.0 a MS-Windows 95 vlastní Technická univerzita v Liberci.

UČEBNICE IPX

VLADIMÍR VAIS



LIBEREC 1998

Úvod

Tato publikace popisuje síťový protokol IPX v prostředí operačních systémů MS-DOS firmy Microsoft Corporation a Novell NetWare firmy Novell, Inc. Text se zabývá danou problematikou z pohledu programátora, který by měl být schopen po přečtení knihy začít sám programovat síťové aplikace s využitím protokolu IPX.

Publikace byla vytvořena jako učební text pro střední školy s rozšířenou výukou informatiky. Její použití je ale samozřejmě možné kdekoliv. Pro sazbu byl použit výborný program LaTeX, dostupný za 0,– (slovy: nula) korun.

Od čtenáře se vyžaduje znalost programování v jazyce Turbo Pascal, alespoň základní znalost systémového programovaní v JSA[1] – jazyk symbolických adres a mírná praxe práce v síti Novell NetWare.

Rád bych také poděkoval všem, kteří napomohli vzniku tohoto textu. Za všechny děkuji alespoň panu RNDr. Pavlu Satrapovi za to, že mě tak dobře naučil programovat v Turbo Pascalu, že mě velmi slušně seznámil s problematikou sítí a za jeho cenné rady ohledně sazby v TeXu. V neposlední řadě musím také poděkovat panu Holečkovi, se kterým jsem sdílel pokoj na koleji, za jeho slušné chování během psaní textu a také řediteli ZŠ Lesní v Liberci Dvořákovi za možnost vydělat si u nich ve škole peníze na slušnější hardware.

Vladimír Vais

Liberec, 23. května 1998

[1] překladač se nazývá Assembler

Obsah

Seznam obrázků

Seznam tabulek

Část I

Počítačové sítě

V následujícím textu se zevrubně seznámíme s počítačovými sítěmi, což se nám bude hodit ve druhé části knihy, která se zabývá programováním s využitím protokolu IPX.

1 Co to je

Česky „počítačová síť", anglicky „Computer Network". Jak už napovídá samotný název, jedná se o systém vzniklý vzájemným propojením počítačů. Takovéto počítače mají oproti svým nezasíťovaným kolegům některé výhody, které zmíníme za okamžik. Podle názvu by se dalo usuzovat, že v počítačové síti budou zapojeny výhradně počítače. Opak je pravdou. Vedle počítačů bývají do sítí obyčejně zapojena i jiná technická zařízení, která mají na starost různé úkoly související se síťovým provozem. Vedle zařízení, která můžeme vidět i mimo síť (např. tiskárny), tu najdeme hlavně různé zesilovače, rozbočovače apod. Vše bude náležitě osvětleno v další části textu.

1.1 Proč se zadrátovat

Počítače zapojené do sítě mají spoustu výhod, které ovšem většinou nebývají na první pohled zřejmé. Kdo v síti nikdy nepracoval, bude se jí nejspíše zuby nehty bránit, avšak jakmile to jednou zkusí, už nikdy se toho nebude chtít vzdát.

K hlavním výhodám sítě patří následující tři:

- sdílení dat
- sdílení prostředků

- spolehlivější systém

1.1.1 Sdílení dat

Sdílení dat je asi největší výhodou ze všech. Jde o to, že jedna data mohou být používána na více počítačích zároveň, avšak uložena jsou jen na jednom místě. Pro nás z toho plyne, že stačí koupit na jeden počítač (server, blíže na straně 17) velké disky, na které vše uložíme. K těmto diskům budou ostatní uživatelé sítě přistupovat, jako kdyby byly připojeny k jejich počítačům. Údržba takového systému je podstatně jednodušší, než kdyby správce musel obejít všechny (může jich být i velmi mnoho) počítače a všude musel nainstalovat novou verzi programu. Mimoto se ušetří ještě spousta peněz za velké harddisky, které by se jinak musely koupit na všechny počítače.

Navíc sdílení umožňuje provozovat systémy, které si o něj přímo říkají. Kupříkladu bankovní síť. Na jednom počítači v centrále jsou všechny účty klientů. Jakákoliv transakce v kterékoliv pobočce, u bankomatu nebo v obchodě se zapisuje po síti do této databáze. Tudíž není možné, aby si člověk, který má na účtě 10 000,– korun, vybral u jednoho bankomatu 3 000,–, u druhého 5 000,– a pak se ještě dobře navečeřel v hotelu Praha za 3 000,– korun. Obdobně to je třeba s rezervací letenek, rekreací apod.

1.1.2 Sdílení prostředků

O jednom případu jsme si již pověděli. Jedná se o sdílení disků. Jeden velký používají všichni. Obdobně se dají sdílet i jiná zařízení, např. pásky, CD-ROMy, tiskárny, ale i výkonné počítače, které mohou zpracovávat náročné úlohy. Například při vytváření počítačových animací do filmu Akumulátor I spolupracovalo mezi sebou po síti sedm počítačů Silicon Graphics.

1.1.3 Spolehlivější systém

Další výhodou sítě je zlepšení spolehlivosti systému. Jedná se o ochranu souborů – spouští-li se program ze vzdáleného počítače, je zpravidla pouze pro čtení, to znamená běžný uživatel ho nemůže smazat, zavirovat či jinak zničit, což by se mu na svém počítači jinak povedlo velmi snadno.

Pak je-li vše na síti, není potřeba, aby na počítačích byly disketové mechaniky ⇒ nikdo nic neodcizí, nikdo nic nezaviruje. Selže-li jedna tiskárna, může se pokračovat s tiskem na jiné. Selže-li jeden počítač, je možné sednout si k jinému a pokračovat v práci. Zálohování dat je také podstatně jednodušší, vše se provede na hlavním počítači, řadový uživatel se o to nemusí vůbec zajímat, vše leží na bedrech správce sítě.

1.2 Rozdělení podle velikosti

Počítačová síť může mít velikost pracovního stolu, místnosti, budovy, areálu, města, státu, kontinentu... Z tohoto důvodu se sítě začaly dělit podle velikosti na sítě **lokální** a **rozlehlé**. V této publikaci se budeme zabývat téměř výhradně sítěmi lokálními.

1.2.1 Lokální sítě

Sítě místního charakteru se označují jako **LAN** (Local Area Network). Obyčejně spojují maximálně několik budov, většinou spíše jen několik místností. Pro lokální sítě je typické, že k propojení jednotlivých počítačů je použito vlastního (nepronajatého) média, tzn. pokud máme např. síť přes dvě místnosti, propojíme jednotlivé počítače třeba pomocí koaxiálního kabelu, namísto abychom využili telefonní rozvod, který už kdysi nainstalovala telefonní firma.

K dalším vlastnostem patří:

- vysoká rychlost
- nízká chybovost
- sdílení prostředků

1.2.2 Rozlehlé sítě

WAN, tedy Wide Area Network. Jedná se o sítě velké, ve kterých je všeho hodně. Provozují se na pronajatých (telefonních) linkách. U sítí tohoto typu je:

- nižší rychlost
- vyšší chybovost
- sdílení prostředků
- komunikace

Existují však i rozlehlé sítě, které v sobě mají úseky mnohem rychlejší a spolehlivější, než jaké najdeme u průměrné lokální sítě. Co se týče sdílení prostředků, tak v lokálních sítích se sdílí hlavně tiskárny a disky, zkrátka přímo hardware.

U rozlehlých sítí bývá spíše zvykem poskytovat hardware ve formě nějaké služby, a to většinou pouze pro čtení. Také nebývá zvykem, aby organizace poskytovala třeba svou tiskárnu pro všeobecné blaho nebo abychom mohli ze svého pokoje přikládat pod kotlem ve spalovně v Čeljabinsku. Existují však i světlé[2] výjimky – kupříkladu firmy, které umožňují, aby si od nich kdokoliv prostřednictvím WWW anonymně odeslal fax nebo elektronický dopis.

[2] nebo možná spíš tmavé

2 Lokální sítě obecně

Nyní nastala ta pravá chvíle, abychom se důkladně seznámili s lokálními sítěmi – zatím ještě poněkud obecněji. Hlavní objekt našeho zájmu, lokální síť Novell NetWare, probereme v části začínající na stránce 21.

2.1 Vlastnosti

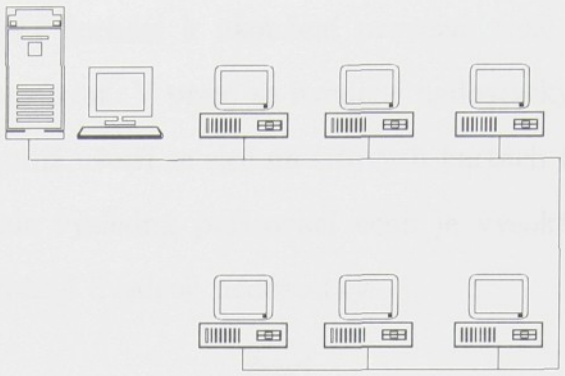
2.1.1 Rozdělení podle topologie

Podle topologie, to jest podle toho, jak jsou počítače mezi sebou propojeny, rozlišujeme následující čtyři druhy sítí.

Sběrnice. (BUS) je velmi oblíbený druh sítě. Dala by se přirovnat např. k vodovodnímu potrubí v panelovém domě. Trubka vede celým domem rovně zdola nahoru a v jednotlivých patrech jsou odbočky pro každý byt. Obdobně tomu je u sběrnice. Uživatelé se napojují pomocí speciálních T-konektorů (vypadá stejně jako vodovodní „téčko") na kabelový segment. Zpráva vyslaná jednou stanicí se šíří po celém vedení ⇒ adresát má data z první ruky, zvídavý špión také.

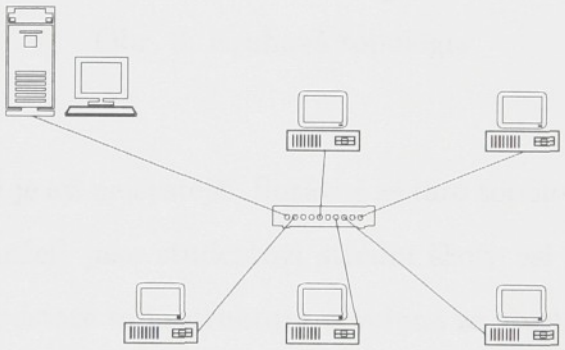
Výhodou sběrnice je, že k propojení je potřeba málo kabelů. Nevýhodou je, že je kabel rozdělen na spoustu kousků, které jsou pospojovány konektory, což jsou potenciální zdroje závad. Při přerušení vedení dochází k odpojení části sítě. Navíc pokud je na kabelu hodně stanic, výkon sítě klesá, protože trvá náhodně (teoreticky i velmi) dlouho, než se stanice „dostane ke slovu".

Hvězda. U topologie hvězda (STAR) je v lokálních sítích pravidlem, že jednotlivé stanice jsou napojeny na **rozbočovač** (koncentrátor, HUB), což je centrum, přes které jdou veškerá data. Často je namísto některého počítače ve hvězdě opět další rozbočovač, čímž vzniká **stromová struktura** (DISTRIBUTED STAR).



Obr. 1: Sběrnicová topologie

Výhodou této topologie je větší spolehlivost – poškození jednoho kabelu vyřadí z provozu většinou jen jednu stanici. Při použití inteligentního středu lze mít oddělený provoz, tzn. data se nešíří na všechny stanice, ale pouze jednou cestou k adresátovi. Nevýhodou je, že výpadek středu je pro celý systém fatální, navíc pro výstavbu je zapotřebí hodně kabelů.

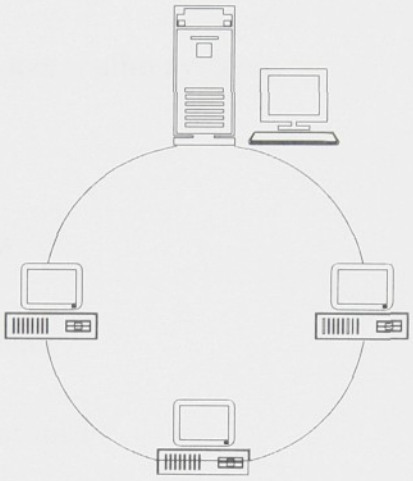


Obr. 2: Hvězdicová topologie

Kruh. Topologie kruhová (RING) je utvořena zapojením stanic za sebou do kruhu. Data se šíří od odesilatele od jedné stanice ke druhé, dokud nedosáhnou určeného cíle. Po síti běhá tak zvaný **token** („pešek"). Která stanice ho má, ta může vysílat, po skončení ho pošle další stanici v kruhu atd., pořád dokola.

K výhodám uvedené topologie patří rychlost – nejsou žádné dohady, kdo bude kdy vysílat, všichni se řídí podle „peška". Síť se dá dobře testovat – pošlu si data sám sobě. Když úspěšně oběhnou kruh a vrátí se, je to v pořádku.

Bohužel při rozpojení dochází k ukončení provozu, také vypnuté stanice je potřeba nějak přemosťovat. V praxi se uvedené nedostatky technicky řeší, leč nic není zadarmo, tudíž ušetří se sice na síťových kartách (jsou levné, protože jsou jednoduché), ale výsledná pořizovací cena je vysoká kvůli technickým zařízením, jež odstraňují uvedené nedostatky.



Obr. 3: Kruhová topologie

Směsice. V praxi je asi nejčastější. Správně se tato topologie nazývá **obecný graf**. To vám (bohužel) jako studentovi střední školy asi mnoho neprozradí. O co tedy jde? V podstatě to je struktura utvořená ze všech výše jmenovaných zapojení.

Uveďme si příklad: Budovou vede kabel (sběrnice), na který je v každém patře napojen koncentrátor, který připojuje jednotlivé kanceláře (hvězda). V každé kanceláři je další koncentrátor, který připojuje jednotlivá pracoviště (hvězda-strom).

2.1.2 Služby sítě

Sítě poskytují pro své uživatele také nějaké služby. Jedná se o:

- předávání zpráv

- přenosy souborů
- konverzace mezi uživateli
- sdílení souborů
- tisk na vzdálených tiskárnách
- vzdálené zavádění operačního systému
- ochrana dat
- účtování služeb sítě
- spojení s jinými sítěmi
- práce na vzdálené stanici

Předávání zpráv. Jedná se o službu, která předává zhruba jednořádkovou textovou zprávu. Adresát MUSÍ být v daném okamžiku přihlášen v síti. Zprávu je možné zaslat konkrétnímu uživateli (Karla Mráčková), skupině uživatelů (Oddělení karbonizace oxohydrátů) nebo všem. Příjemce zprávu po obdržení „odklepne“ zvolenou kombinací kláves, dokud to neudělá, je jeho systém blokován (existují výjimky, např. v MS-Windows je možné okno se zprávou odložit na pozadí a dál si hledět své práce).

Přenosy souborů Jedná se o klasický přenos souborů, který se děje prostřednictvím sítě. Je to stejné, jako kdybychom něco uložili na disketu, a tu pak odnesli kolegovi. Nicméně po síti to je mnohem pohodlnější a navíc to bývá i rychlejší. Sítě umožňují přenosy souborů mezi stanicemi a mezi stanicemi a serverem.

Konverzace mezi uživateli. Tahle služba umožňuje, aby si dva aktivní uživatelé mohli spolu „povídat". Celá věc funguje tak, že každý z účastníků má přidělenu jednu polovinu obrazovky, na kterou může psát, na druhou polovinu píše jeho diskusní partner. Služba se označuje jako **TALK**.

Obdobou je služba **CHAT**, která dovoluje konverzovat většímu počtu osob. Na obrazovce má každý část, ve které si může připravit svůj text. Po odeslání se objeví (včetně jména autora) všem zúčastněným.

Příklad:

Tady je malá ukázka obrazovky uživatele Egona:

```
*** Helga entered.
Helga: Nazdar, tak jsem tady!
Udo: Ahoj, víš, jak se řekne maďarsky potápěč?
Helga: To nevím.
Udo: Čumízgumy. Cháááááááááááááá!
Hans: Ha ha ha!
Helga: Hmm, dobrý. A víš, jak se řekne maďarsky zubař?
Udo: Ne.
Egon: Ale já jo. Denta aparátoš.
Helga: To nemá cenu, já nehraju!
*** Helga has been disconnected.
Egon: Škoda
------------------------------------------------------------
Egon: Mě už to nebaví, jdu radši něco dělat.
```

Sdílení souborů. Toto již bylo vysvětleno v části 1.1.1 na straně 7. K již uvedenému dodejme: Umístění velkého množství dat na jednom místě má kromě všech uvedených výhod i jednu nevýhodu. Tou je znepřehlednění. Nalezení

určitého souboru pak připomíná hledání jehly v kupce sena. Naštěstí i na tohle bylo pamatováno. Velký disk na serveru je možno díky tzv. **mapování** předkládat uživatelům po kouskách. Libovolný adresář na serveru je možno předkládat jako kořen fiktivního disku na uživatelově počítači. Nejlepší bude, uvedeme-li si příklad:

Příklad:

Ve firmě jsou oddělení DTP, Grafika a Písárna. V prvním oddělení se používá výhradně program LaTeX, ve druhém pouze Neopaint a ve třetím jen T602. Pokud se do sítě přihlásí kdokoliv z oddělení DTP, bude mít na disku `P:` pouze adresář LATEX, zaměstnanci z Grafiky tam budou mít jen NEOPAINT a písařky jen T602.

Na stejném principu se provádí např. mapování domácích adresářů jednotlivých uživatelů. Domácí adresář je místo na serveru, které má každý většinou jen pro sebe, k ostatním se podívat nemůže, stejně tak oni nemohou koukat k němu. Správce sítě rozhodne, že domácí adresář bude mít každý namapován třeba na disk `X:`. Přihlásí-li se uživatel OSKAR, bude mít na disku `X:` namapováno `\HOME\SKLAD\OSKAR`. Přihlásí se BOZENA. Ta bude mít disk `X:` namapován jako `\HOME\VEDOUCI\BOZENA`.

Tisk na vzdálených tiskárnách. Tiskárna je velmi používané zařízení. Nicméně většinou žádný zaměstnanec není tak produktivní, aby tiskl v jednom kuse, tudíž jednu tiskárnu může používat více lidí. Pokud nejsou počítače v síti, existují dvě možnosti. Buď si zaměstnanci tiskárnu půjčují – velmi nepraktické, nebo chodí tisknout k počítači s tiskárnou – také nepraktické. Pokud síť máme, máme vyhráno. Tiskárna se připojí k libovolnému počítači nebo přímo na síťové vedení. Na serveru se vytvoří tzv. **fronta**, do které uživatelé posílají své tiskové úlohy (joby), které se pak postupně tisknou v pořadí, v jakém do fronty

dorazily. Pro uživatele se nic nemění, jejich počítače se tváří, jako kdyby měly tiskárnu opravdu připojenou, např. na port LPT 1.

Vzdálené zavádění operačního systému. Tato služba umožňuje zavedení operačního systému na stanici ze sítě. Používá se buď v případě, kdy počítač nemá žádné disky (třeba aby se nedalo ze sítě nic vynášet na disketách), ze kterých by se dal zavést operační systém, nebo v případě, že správce sítě chce mít kontrolu nad konfigurací jednotlivých stanic. Tímto se zároveň znemožní uživatelům, aby poškodili startovací soubory – tohle se velmi hodí ve školách a různých školících střediscích, kde si uživatelé s oblibou „hrají" tak, že příště už počítač nenastartuje.

Ochrana dat. Sítě většinou disponují prostředky, které chrání data jednak před úmyslným zničením (ale i zneužitím), dále před neúmyslným poškozením.

Ochrana proti cílenému útoku je v sítích realizována systémem uživatelských jmen a hesel. K důležitým informacím na síti má přístup vždy jen část lidí, kteří se přihlašují se zadanými jmény a hesly. Pokud nájezdník nezná příslušné přihlašovací jméno a heslo, má smůlu.

Chránit data proti náhodnému zničení je druhý úkol sítě. V praxi se používají metody **zrcadlení disků**, všechna data se ukládají duplicitně na různé disky. Pokud jeden selže, použije se druhý. Další okamžik, kdy se mohou data poškodit, je při výpadku napájecího napětí. Ochranou jsou **zdroje nepřetržitého napájení** (UPS), které dokáží po nějakou dobu držet systém pod napětím. Pokud nedojde ve stanovené době k obnovení dodávky proudu v síti, je provoz sítě automaticky korektně ukončen, aniž by došlo k nějakým ztrátám informací.

Účtování služeb sítě. Uživatel má přidělen určitý počet bodů, které se mu odečítají za využívání sítě. Platí se za čas strávený na síti, za uložení dat na

serveru, za služby sítě apod. Když uživatel přidělené body vyčerpá, koupí si u správce sítě další. Ve většině případů se tento systém nevyužívá nebo se využívá pouze k statistickým účelům o využití sítě.

Práce na vzdálené stanici. Tato služba je standardně k dispozici na počítačích s operačním systémem typu Unix. Uživatel se přihlásí ze své stanice pomocí speciálního programu na unixový server a pracuje na něm stejně, jako kdyby seděl přímo u něj. U ostatních sítí to nebývá standardně k dispozici. Existují ale programy, které to umožňují. V DOSu je to známý „LAN Assistant", pod Windows 95 lze použít např. „Laplink 95" nebo „McAfee Remote Access". Pro běžnou práci nejsou příliš vhodné pro přílišnou pomalost (zvláště Laplink přes modem), hodí se k občasným zásahům, např. když chce správce někomu pomoci v nesnázích.

2.2 Síťový hardware

2.2.1 Druhy stanic

V síti existují dva druhy stanic. Jsou to **servery** a **pracovní stanice**.

Server. Jedná se o stanici, která zajišťuje chod sítě a poskytuje služby pracovním stanicím. Z toho důvodu bývá server zpravidla velmi výkonný počítač s velkými disky a pamětí RAM. V síti může být serverů několik (tzv. multiserverové prostředí). V takovém případě může každý server poskytovat jiné služby – tisk (print server), elektronickou poštu (mail server), zpracování dat (application server) apod. Většinou nebývá zvykem, aby uživatel pracoval přímo na počítači, který realizuje server. Buď to není dost dobře možné (Novell NetWare – lze používat jen příkazy serveru) nebo to možné je (MS-Windows NT Server, Unix), ale kvůli zatížení serveru je práce pomalá a nepříjemná. Tohle samozřejmě neplatí vždy, pokud je server dostatečně technicky vyba-

ven, projeví se na něm citelné zpomalení až při současném připojení desítek až stovek uživatelů zároveň. Typickým příkladem je BBS na TU v Liberci.

Pracovní stanice (Workstation). Stanice, na které pracuje uživatel. Při své práci využívá služeb síťových serverů. Práce na stanici v síti se příliš neliší od práce mimo síť, pokud není síť využívána. Při velkém využívání sítě (pošta, talk, vzdálený tisk...) je rozdíl oproti nesíťovému provozu značný.

2.2.2 Ostatní železo

Pro neznalé: „Železem" se zde rozumí **hardware**, tedy něco, na co si lze sáhnout (počítače, kabely, tiskárny, diskety apod.). Podobný termín **software** naopak označuje programové vybavení.

Síťové karty. Síťové karty jsou zásuvné karty do počítače, které na sobě mají konektory pro připojení na síťové vedení. Tyto karty se obyčejně připojují do slotů sběrnice ISA nebo PCI – obdobně jako zvukové, grafické, modemové nebo jiné karty. Prostřednictvím síťové karty počítač komunikuje se sítí.

Spojovací vedení. Počítače mohou být pospojovány mnoha způsoby, jedná se zejména o tyto druhy médií:

- **Koaxiální kabel** se vyznačuje příznivou cenou, snadným konektorováním a slušnou odolností proti rušení elektromagnetickým polem. Používá se u sběrnicové topologie.
- **Kroucená dvoulinka** má obdobné vlastnosti jako koaxiální kabel. Nedělají se z ní ale odbočky jako z koaxiálního kabelu, spojují se s ní vždy pouze dvě místa, např. stanice a koncentrátor.
- **Optický kabel** je nejmodernějším, ale také hodně drahým spojovacím médiem. K vlastnostem patří absolutní odolnost proti rušení elektro-

magnetickým polem, nemožnost odposlechu, vysoká přenosová rychlost. K nevýhodám patří poměrně obtížné a drahé konektorování.

- **Vzduch** je zdarma[3]. Bohužel jako spojovací médium patří k nejdražším. Používají se různé mikrovlnné, infračervené nebo rádiové spoje. Vysílače a přijímače nepatří zrovna k nejlevnějším, navíc je pro provoz nutná téměř vždy přímá viditelnost od vysílače k přijímači – toto se dá obejít použitím mezičlánku (třeba užitím telekomunikační satelitní družice).

Repeater. Jedná se v podstatě o zesilovač signálu. Zařízení je připojeno k oběma kabelovým segmentům. Vše, co repeater zachytí na jednom kabelu, vyšle i do kabelu na druhém konektoru a naopak. Takto vyslaný signál je silnější než původní, který mohl přijít z velké vzdálenosti náležitě zeslabený a zkreslený odporem vedení. Zpoždění na repeateru je 1 bit. V cestě mohou z časových důvodů stát maximálně dva repeatery.

Repeatery se vyrábějí ve dvou variantách.

- **Local** (místní) varianta je pouze jedno zařízení, do kterého jsou zapojeny oba segmenty sítě.

- **Remote** (vzdálená) varianta používá dvě zařízení, každé je připojeno k jednomu segmentu. Mezi sebou jsou spojena mikrovlnně, infračerveně apod.

Bridge. Můstek je zařízení podobné repeateru. Stejně jako on zesiluje signál, navíc ale umí odlehčit síti. Na výstup neposílá vše, co přijme na vstupu, ale jen to, co tam patří (data určená stanicím za můstkem ležícím). Můstky se samy učí, podle adres odesilatelů poznají, kde která stanice leží. Pokud můstek dostane neznámá data, pošle je dál. Můstky jsou zcela transparentní, nikdo neví, že tam jsou. Vyrábějí se v provedení local a remote.

[3] dokonce i u benzínové pumpy

Router. Jedná se o zařízení ještě inteligentnější. Odděluje jednotlivé podsítě (části adresního prostoru). Provoz je výpočetně náročný. Standardem jsou produkty CISCO, které jsou mimořádně kvalitní, čemuž odpovídá i jejich cena – desetitisíce až milióny korun.

3 Novell NetWare letecky

Sítě Novell NetWare patří v současné době k tomu nejlepšímu, co lze mezi síťovými operačními systémy nalézt. Vedle vysokého počtu kvalitních služeb nabízejí tyto sítě i vysokou spolehlivost, výbornou ochranu dat a snadnost připojení k jiným sítím. Pro sítě Novel NetWare existuje obrovské množství programového vybavení pro stanice i pro servery. Sítě se liší svou cenou podle počtu poskytovaných služeb a podle maximálního počtu připojených uživatelů. Zde je nutno podotknout, že oproti serveru Windows NT tu NetWare trochu zaostává. NetWare se prodává ve verzích od 5 do asi 1 000 uživatelů. U verze 4 si lze počty uživatelů kumulovat, čili 2x NW pro 1 000 uživatelů = 1x NW pro 2 000 uživatelů. Windows NT server podporuje až 10 000 (!) uživatelů.

Není ale všechno zlato, co se třpytí. NT-server sice koupíte levněji, leč abyste dosáhli stejného výkonu, budete muset ještě investovat nemalé peníze do hardwaru a kromě toho správa NT-serveru je několikanásobně obtížnější než správa Novellovského serveru. Nástrahy číhají na každém rohu, např. jedním neopatrným klepnutím myši můžete sebrat všem uživatelům práva v jejich domácích adresářích. Jistě budou rádi, minimálně tak jako vy, protože to budete muset ručně vrátit do původního stavu. Uvedený případ je z jednoho nejmenovaného libereckého podniku, kde nakonec po nervovém zhroucení správce sítě přešli z NT na Linux s NetWare emulátorem.

3.1 Trocha historie nikoho nezabije

NetWare v dnešní podobě je výsledkem mnohaleté práce týmů odborníků. Během této doby bylo na svět vypuštěno mnoho verzí. V tabulce je stručně uvedeno, jak postupoval vývoj. V současné době jsou asi nejrozšířenější verze 4.x.

ELS NetWare Level I.	Nejjednodušší varianta, najednou mohou pracovat jen 4 uživatelé, mnoho toho neumí.
ELS NetWare Level II.	Umožňuje práci 8 lidí, spolupracuje s významnými prostředky sítě, vhodný pro malé podniky.
Advanced NetWare	Podporuje až 100 uživatelů, byl velmi rozšířený.
SFT NetWare	Podporuje až 250 lidí, má již zabudovány prostředky ochrany dat.
NetWare 286	Přímý následník, zůstává 16-bitový, poskytuje nové služby, verze 2.2x velmi úspěšné.
NetWare 386	Již 32-bitový, potřebuje procesor alespoň 80386, vyšší výkon, více služeb, oblíbené verze 3.11 a 3.12, od verze 4 podpora více serverů v síti, domén, Internetu a Intranetu, vypuštěny některé již překonané vlastnosti.

Tabulka 1: Verze Novell NetWare

3.2 Programové vybavení stanice

Aby stanice mohla komunikovat se sítí, je zapotřebí, aby na ní byly nainstalovány ovladače pro síťový provoz. V nejjednodušším případě jsou zapotřebí dva programy: IPX.COM a NETX.COM. V praxi se ale většinou používají minimálně tři. Problém je v tom, že pokud chceme používat jen dva programy, IPX.COM se musí vygenerovat pro každou síťovou kartu zvlášť. To je nepraktické, proto se používá ještě třetí soubor – **ovladač karty** (většinou ho koupíte spolu s kartou). V takovém případě je možno použít jeden univerzální IPX.COM pro všechny stanice.

IPX.COM umožňuje komunikaci se sítí prostřednictvím protokolů IPX a SPX.

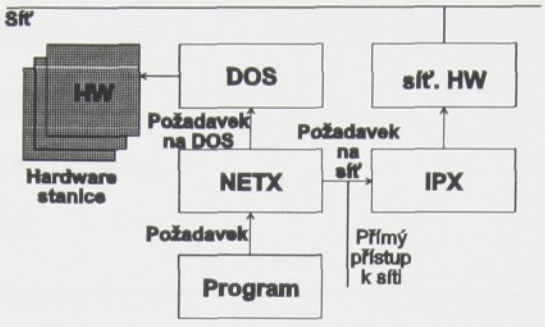
NETX.COM je tzv. **shell**. Jeho starostí je zachytávat veškeré požadavky od uživatele a běžících programů a směrovat je tam, kam patří, tj. buď operačnímu systému na stanici (naformátování diskety) nebo síti (zaslání dopisu). Mezi jeho úkoly patří např. i již dříve zmiňovaný tisk. Pokud není port přesměrován na síťovou frontu, propouští NETX data přímo na připojenou tiskárnu, pokud přesměrován je, odchytává data a posílá je po síti do fronty. Jelikož je NETX poměrně velký rezidentní program (cca 40 KB[4]), dodává se i ve verzích pro EMS a XMS (EMSNETX.EXE, XMSNETX.EXE).

Pořadí nahrávání ovladačů do paměti je následující:

1. ovladač karty (NE2000.COM, 3C59X.COM apod.)
2. ovladač IPX/SPX (IPX.COM)
3. NetWare Shell (NETX.COM, EMSNETX.EXE, XMSNETX.EXE)

[4]Pozor! KB = 1 024 B, kB = 1 000 B

K dalším používaným programům patří emulátor **NETBIOS.EXE**. Jeho použití je nezbytné v případě provozování programů, které vyžadují rozhraní NetBIOS. Těchto programů naštěstí není mnoho, proto je možné, že tento emulátor ani nebudete nikdy potřebovat.



Obr. 4: Funkce modulů IPX a NETx

ODI Shell (Open Data-Link Interface) je prostředek, který zajišťuje, že je pracovní stanice schopna zpracovávat více než jeden komunikační protokol. Navíc je při použití ODI-ovladačů spojení se sítí rychlejší. Nevýhodou je větší obsazení paměti rezidentními programy. Základem všeho je program **LSL.COM**, který realizuje vlastní možnost komunikace s několika protokoly. Při zavádění do paměti se nejprve nahraje ovladač karty, potom LSL.COM. Dále pro Novell NetWare IPXODI.COM a NETX.COM, dále třeba ovladače pro TCP/IP, AppleTalk apod.

VLM moduly v současné době nabízejí nejmodernější a nejefektivnější možnost připojení stanice k síti z MS-DOSu. Místo programu NETX se nahrává program VLM, který má v konfiguračním souboru uvedeny moduly, které má umístit do paměti, tzn. je možnost vybrat si pouze ty, které na stanici SKUTEČNĚ potřebujeme a tím minimalizovat paměťové nároky síťových ovladačů.

Příklad:

Příklad demonstruje postupné zavádění ovladačů při použití VLM, jak to má

na svém počítači uděláno správce sítě TU Liberec pan Petr Adamec.

```
lh c:\liane\lsl
lh c:\liane\3c59x
lh c:\liane\odipkt30 0 96
lh c:\liane\winpkt 0x60
lh c:\liane\ipxodi
lh c:\windows\odihlp
lh c:\liane\vlm
```

3.3 Typy uživatelů

Jak už to v životě bývá, někdo může všechno, jiný skoro nic. Obdobně tomu je i v Netwaru. Po nainstalování systému existují dva uživatelé: **Supervisor** a **Guest**.

Supervisor je pánem serveru. Smí skoro všechno (co nesmí on, smí už leda samotný operační systém). Má za úkol tvořit nové uživatele, instalovat programy, tiskové fronty apod.

Guest je uživatel, který ani nemá nastavené heslo. Bývá zvykem, že nemá valná práva, slouží pouze pro „navštívení" sítě. Často je tento účet zamčený, aby nebylo možno v síti pracovat anonymně, koukat, jaké účty jsou v síti zavedeny, posílat (drzé) zprávy ostatním apod.

Dále v síti existují uživatelé nejrůznějšího druhu a nejrůznějších oprávnění. Je-li síť velká, nestačí Supervisor na všechno. Mívá proto několik pomocníků, kteří mají buď oprávnění jako on, či jen taková, která jsou potřeba pro jejich činnost (správa uživatelů, správa softwaru, správa tisku apod.). K nejpočetnější skupině patří **běžní uživatelé**, kteří si dělají na síti práci nijak nesouvisející s její správou, pouze využívají, co jim správci nachystali.

Uživatelé mohou být v tzv. **skupinách**, čímž se správa systému podstatně zjednodušuje. Do adresáře nebo k tiskové frontě stačí přidělit práva skupině, tím tam dostali právo i všichni členové této skupiny. Skupiny mají názvy, které vysvětlují důvod jejich vytvoření (TISK_LASER, RCONSOLE, NOTICE_ADMIN, TRESTANCI) nebo názvy skupin nějak k sobě patřících lidí (TEACHERS, STUDENTS, AIESEC).

3.4 Přístupová práva v NW 3.1x

Novell NetWare umožňuje přidělovat jednotlivým adresářům a souborům na síťových discích tzv. práva (rights). Použitelná práva **S**, **R**, **W**, **C**, **E**, **M**, **F**, **A** jsou popsána v přehledné tabulce na následující stránce.

V tabulce se vyskytuje pojem **atributy**. Z operačního systému MS-DOS známe čtyři: Archive (určen k archivaci), System (systémový), Hidden (skrytý) a Read Only (jen pro čtení). V Netwaru jsou k dispozici ještě další. V tabulce dále v textu jsou všechny popsány. V manuálu NetWare se uvádí ještě atribut **I** – Indexed. Tento atribut je NetWarem přidělován automaticky souborům, které přesáhnou určitou velikost. Velký soubor se pak na serveru ukládá jiným způsobem, aby se v něm dalo rychleji hledat.

K tabulce ještě dodejme: Atribut Transactional zajišťuje, že pokud operace se souborem nebyla korektně ukončena, je navrácen do původního stavu. Takto je např. chráněna systémová databáze Bindery. Jestliže při zapisování dojde třeba k výpadku proudu, je po novém nastartování systému soubor vrácen do stavu, v jakém se nacházel před započetím transakce. Atribut Execute Only se používá k ochraně souboru před zcizením, zkoumáním nebo třeba zavirováním. Nastavit tento atribut může každý s potřebnými právy (nevěřte, že jen Supervisor, jak se občas uvádí), odstranit nikdo. Takto chráněný soubor lze pouze spouštět. Bohužel není možno takto chránit soubory, které ze sebe čtou ještě během chodu, jde třeba o m602.exe od Manažeru602 nebo o bp.exe od

Právo	**Význam pro soubor**	**Význam pro adresář**
Supervisory	všechna práva	všechna práva k adresáři i podadresářům
Read	právo otevírat a číst obsah	právo otevírat a číst obsah souborů
Write	právo otevírat a zapisovat do něj	právo otevírat soubory a zapisovat do nich
Create	právo obnovit smazaný soubor	právo vytvářet soubory a podadresáře
Erase	právo smazat	právo mazat soubory a podadresáře
Modify	právo přejmenovat a měnit atributy (nikoliv obsah)	právo přejmenovávat soubory a podadresáře a měnit jim atributy
File Scan	právo vidět soubor v adresáři	právo vidět obsah adresáře
Access Control	právo měnit práva mimo S	právo měnit práva mimo S

Tabulka 2: Práva NW 3.1x

Archive needed	soubor ještě nebyl archivován
Copy Inhibit	soubor nelze kopírovat na počítačích Macintosh
Delete Inhibit	soubor (adresář) nelze smazat
Hidden	soubor (adresář) není vidět, např. příkazem DIR
Purge	soubor (adresář) nelze po smazání obnovit
Rename Inhibit	soubor (adresář) nelze přejmenovat
Read **a**udit	k ničemu, historické důvody
Read **o**nly/**R**ead **w**rite	Rw – pro čtení i zapisování, Ro – jen pro čtení
Shareable	soubor je sdílitelný, může ho otevřít více lidí najednou
System	systémový soubor
Transactional	soubor je chráněn systémem sledování transakcí TTS
Write audit	k ničemu
EXecute Only	soubor lze jen spouštět
Indexed	indexovaný soubor

Tabulka 3: Atributy v NetWare 3.1x

Borland Pascalu. Atribut se hodí pro často používané programy jako MAP, LOGIN nebo SYSCON, které lze takto chránit před zavirováním Supervisorem (smutný případ, bohužel nikoli ojedinělý). Pro pořádek ještě dodejme, že ochrana je plně v rukou programu NETX. Pokud si ho někdo upraví, může si ze sítě potom odnášet i takto chráněné soubory.

Nyní ještě několik poznámek k právům. Podotýkám, že tato publikace není příručkou Netwaru, ale knihou o programovaní s IPX. Část o sítích je zde v podstatě pouze něco navíc pro alespoň základní orientaci ne příliš znalého čtenáře. Pro detailnější informace doporučuji např. publikaci [3]

Na síťových discích se práva **dědí** z adresáře, kde byla přidělena, do všech jeho podadresářů. Problematika práv není triviální. Při zjišťování aktuálních práv v adresáři se musí brát v úvahu práva uživatele, práva skupin, jichž je členem, a stav tzv. **masky zděděných práv** (Inherited Rights Mask – IRM), která omezuje všechny uživatele (nemající právo S) s právy do adresáře, ke kterému je nastavena. Pro podrobné vysvětlení opět doporučuji knihu [3], my si zde uvedeme jen to nejzákladnější. IRM si je možno představit jako jakési „sítko“, kde každému z práv SRWCEMFA odpovídá jeden otvor, který je buď otevřen (povoleno) nebo uzavřen (nedovoleno). Po vyhodnocení práv uživatele v daném adresáři se jeho teoretická zděděná práva je „prosijí“ sítem IRM, a to, co „propadne“, jsou jeho aktuální práva. IRM se uplatňuje jen při dědění. Pokud má uživatel přidělena práva pro zkoumaný adresář nebo soubor, mají přednost.

Příklad:

Uživatel HANS je členem skupiny A, která má v adresáři `P:\PROGS` práva `[ RWC FA]`, a skupiny B, která tam má právo `[ M ]`. IRM je v adresáři `P:\PROGS\UTIL` nastavena na `[SR C MFA]`. Jaká práva má tedy HANS v adresáři `P:\PROGS\UTIL`?

```
          A:   [ RWC  FA]
          B:   [      M ]
--------------------------
 HANS celkem:  [ RWC MFA]
       MASKA:  [SR C MF ]
--------------------------
HANS opravdu:  [ R C MF ]
```

Pozor! Pokud má uživatel právo A a maska ho propouští, má zároveň právo měnit masku, což je nebezpečné.

3.5 Základní adresáře Novellu

Síťový operační systém Novell NetWare má po nainstalování několik adresářů. Všechny jsou vytvořeny na svazku SYS. Jedná se o adresáře SYSTEM, PUBLIC, LOGIN a MAIL.

SYSTEM je výlučně systémový adresář, běžný uživatel jeho obsah nikdy neuvidí. Nachází se v něm soubory vlastního operačního systému a programy určené Supervisorovi (RCONSOLE, PAUDIT, SECURITY atd.).

PUBLIC je adresář pro veřejné používání. Obsahuje všechny soubory pro realizaci příkazů NW. V jeho podadresářích často bývají různé verze OS MS-

DOS, podpora pro tiskárny apod. Do tohoto adresáře mají přístup všichni uživatelé.

LOGIN je speciální adresář, který je jako jediný viditelný ještě před přihlášením do sítě. Obsahuje obyčejně programy LOGIN a SLIST, aby se bylo možno přihlásit do sítě.

MAIL je určen především pro elektronickou poštu. Všichni uživatelé v něm mají pouze právo C (aby mohli vytvářet dopisové soubory, ale aby nic neviděli). Každý uživatel má v tomto adresáři svůj podadresář, který má jméno shodné s uživatelovým identifikačním (hexadecimální číslo, které je uživateli automaticky přiděleno v okamžiku vytvoření účtu). V tomto adresáři je také uložen Login Script.

3.6 Několik užitečných příkazů

Na následujících stránkách se seznámíme s několika užitečnými příkazy, které lze užívat na stanici připojené do sítě. Nebude se jednat ani zdaleka o všechny příkazy, které Novell NetWare nabízí. U většiny příkazů se po spuštění s parametrem `/?` zobrazí stručná nápověda. Vyzkoušejte si to!

LOGIN – přihlašuje k zadanému serveru

`LOGIN [volba] [server/[uživatel]]`

Jako volby je možno použít následující:

- **/S** – neprovede se systémový a uživatelský Login script (viz následující odstavec), jako parametr se zadává jméno náhradního scriptu; používat s volbou /N
- **/N** – provede se zvláštní Login script, aniž by došlo ke zrušení stávajícího přihlášení k serveru; používat s volbou /S

- /C – po zadání hesla smazat obrazovku

Pokud se program spustí bez zadání jména serveru a uživatele, je nutno je na vyžádání programu vložit. Při použití programu LOGIN dochází ke zrušení všech připojení k síťovým serverům.

V předcházejícím odstavci se vyskytl termín **Login script.** Jedná se o soubor speciálních příkazů, které se provedou po přihlášení uživatele do sítě. Společný script pro všechny uživatele je uložen v souboru `NET$LOG.DAT` v adresáři `SYS:PUBLIC`. Dále může mít každý uživatel ještě vlastní script v souboru `SYS:MAIL/USER_ID/LOGIN`. Tento soubor je přístupný přes program SYSCON, kde je také možno po stisku klávesy F1 obdržet nápovědu.

Příklad:

```
F:\LOGIN>login salome/vladimir.vais
Enter your password:
You are user VLADIMIR.VAIS on server SALOME.
```

ATTACH – připojení k dalšímu serveru.

`ATTACH [server[/uživatel]]`

Program slouží k připojení uživatelovy stanice ke zvolenému serveru. Pokud jsou zadány oba parametry, program se zeptá pouze na heslo, je-li potřeba, pokud některé parametry chybí, program si je vyžádá. Program neruší již existující připojení k jiným serverům (narozdíl od programu LOGIN).

Příklad:

Pavel Satrapa se připojuje k serveru Herodes.

```
X:\>attach herodes/pavel.satrapa
```

```
Password:
You are attached to server HERODES.
```

LOGOUT – odhlášení od serveru

```
LOGOUT [server]
```

Pokud je spuštěn bez parametru, provede odhlášení od všech serverů, ke kterým je uživatel připojen. S parametrem provede odhlášení pouze od specifikovaného serveru. Příkaz by měl být použit vždy před ukončením práce na stanici.

Příklad:

```
X:\>logout aquila
VLADIMIR.VAIS logged out from server AQUILA connection 4.
Login time:   Sunday  March  1, 1998  6:56 am
Logout time:  Sunday  March  1, 1998  5:14 pm
```

SETPASS – změna přihlašovacího hesla

```
SETPASS [server] [/jméno]
```

Program umožňuje změnit heslo uživatele pro zadaný server. Celý proces probíhá formou dialogu, ve kterém je uživatel nejprve dotázán na své nynější heslo, potom dvakrát (pro kontrolu) na nové heslo. Je-li uživatel přihlášen k více serverům, nabídne mu program změnu hesla také na nich. Parametr `server` se používá, pokud chce uživatel změnit heslo jen na jednom určitém serveru. Parametr `/jméno` může používat jen Supervisor (a jemu rovní), a to pro změnu hesla určitému uživateli.

Malá poznámka k heslům. Heslo do sítě Novell NetWare může být max. 127 znaků dlouhé. Nezáleží na malých a velkých písmenech. V hesle nepoužívejte

češtinu a různé řídící znaky (ALT + číslo). Použití čísel je vhodné, zadávejte je vždy na numerické části klávesnice s rozsvícenou kontrolkou Num Lock. Minimální délka hesla je standardně 5 znaků, Supervisor ale může tuto hodnotu změnit. Aktuální hodnotu zjistíte v programu SYSCON nebo tak, že příkazu `SETPASS` zadáte nějaké kraťoučké heslo, on vám vyhubuje a napíše, že heslo musí mít minimálně X znaků.

Čeho se vyvarovat při výrobě hesla? Nepoužívejte jména, rodná čísla, čísla OP, SPZ, běžná slova (kocka, skola, rybnik) apod. Měli byste si vymyslet heslo, které si nikdo nedomyslí, ani když zná některá písmena. Jedním z řešení je vymyslet si nesmyslné slovo, např. hubafobrherdofix. Lepším řešením je složit heslo z něčeho, co znáte: **A**zore, **A**zore, **A**zore! **Z**avolala **třikrát**. **P**ravil **J**iří **L**ábus. ⇒ `aaa!z3xpjl` bude to pravé heslo.

Příklad:

```
X:\>setpass
Enter old password for TYTO/VLADIMIR.VAIS:
Enter new password for TYTO/VLADIMIR.VAIS:
Retype new password for TYTO/VLADIMIR.VAIS:
Password for TYTO/VLADIMIR.VAIS has been changed.

AQUILA/VLADIMIR.VAIS
COMENIUS/VLADIMIR.VAIS
OTUS/VAIS

Synchronize passwords on these file servers
with TYTO/VLADIMIR.VAIS? (Y/N) Y
Password for AQUILA/VLADIMIR.VAIS has been changed.
Password for COMENIUS/VLADIMIR.VAIS has been changed.
```

```
Password for OTUS/VAIS has been changed.
```

SLIST – seznam serverů na síti

SLIST [server]

Příkaz vypíše seznam serverů v síti. U každého serveru je uvedeno jméno, číslo sítě, ve které leží, fyzická adresa a zda je k němu uživatel přihlášen. Při použití parametru se zobrazí informace jen o zvoleném serveru.

Příklad:

```
X:\>slist
Known NetWare File Servers     Network   Node Address  Status
--------------------------     -------   ------------  ------
ALADDIN                        [93E68468][           1]
AQUILA                         [F3E6010A][           1]Attached
CATBERT                        [93E61071][           1]
COMENIUS                       [      18][           1]Attached
HAWK                           [       9][           1]
KNIHA                          [  111194][           1]
KOMA                           [24E556CD][           1]
KRT                            [34E88ED0][           1]
KTK                            [    FFF1][           1]
OTUS                           [93E6106D][           1]Attached
TYTO                           [    FFF8][           1]Default
VULTUR                         [93E61064][           1]

Total of 12 file servers found
```

USERLIST – informace o přihlášených uživatelích

`USERLIST [server/][jméno] [/A]`

Program zobrazuje seznam přihlášených uživatelů s informacemi o době jejich přihlášení, celém jméně a čísle spojení se serverem. Při použití parametru `/A` (all) se zobrazí ještě další informace, adresa stanice a adresa sítě, kde stanice leží. Pomocí dalších parametrů lze získávat informace jen o určitých uživatelích a určitých serverech. Lze používat žolíkový znak *.

Příklad:

```
X:\>userlist
User Information for Server TYTO
Connection  User Name       Login Time
----------  --------------  ------------------
     1      VLADIMIR.VAIS    3-01-1998  5:45 am
     2    * VLADIMIR.VAIS    3-01-1998  5:55 am
     7      RADEK.VOTRUBEC   2-28-1998  5:02 pm
    11      VITEZSLAV.SEM    2-28-1998  7:48 pm
    13      PETR.KRETSCHME   2-28-1998  1:34 pm
    29      OTUS             2-26-1998  4:30 pm
    68      EVZEN.AUGUSTA    2-23-1998  7:44 am
```

WHOAMI – zobrazuje informace o uživateli

`WHOAMI [server] [volby]`

Who am I? Kdo jsem já? Bez parametrů příkaz vydá informace o přihlášeném uživateli a serveru, ke kterému je připojen. S parametrem `server` se zaměří jen na zvolený server.

Zajímavější jsou další volby, které rozhodně nejsou neužitečné:

- **/S** – komu je uživatel roven na úrovni oprávnění (většinou skupinám, v nichž je členem)
- **/G** – jména skupin, jichž je uživatel členem
- **/W** – informace o manažerovi skupiny (nastavuje se v SYSCON)
- **/R** – efektivní práva uživatele
- **/SY** – informace o systému
- **/A** – všechny informace

Příklad:

```
F:\>whoami /a
You are user SUPERVAIS attached to server SALOME, connection 3.
Server SALOME is running NetWare v3.11 (20 user).
You are a workgroup manager.
Login time: Friday  February  27, 1998  3:09 pm
You are security equivalent to the following:
    EVERYONE (Group)
    SUPERVISOR (user)
    UCITELE (Group)
    GROUPMAIL (Group)
You are a member of the following groups:
    EVERYONE
    UCITELE
    GROUPMAIL
[SRWCEMFA]  SYS:
[SRWCEMFA]  APPS:
Server SALOME is not in a Domain.
```

FILER – umožňuje pracovat se soubory.

`FILER`

Interaktivní program pro práci se síťovými disky. Umožňuje nastavovat práva, měnit masku zděděných práv, měnit vlastníky souborů a adresářů, měnit atributy, zjišťovat informace o souborech, kopírovat, mazat atd. Program rozhodně stojí za vaši pozornost. Navíc občas se stává, že nějaký soubor na síťovém disku nelze z nějakého (neznámého) důvodu smazat, Norton, M602 a jim podobní většinou selžou, Filer to většinou zvládne.

VOLINFO – informace o discích serveru

`VOLINFO`

Program zobrazuje informace o discích (velikost a volné místo v kilobajtech a blocích). Ovládá se pomocí menu se dvěma položkami: Change Server a Update Interval. Program je bohužel velmi oblíben různými záškodníky. Po jeho spuštění se méně výkonné servery lehce vytíží na maximum a síť se stává velmi pomalou pro všechny uživatele, což u „někoho“ způsobuje stav blaženosti. Velice markantní to je hlavně tehdy, má-li server velké disky a hodně uživatelů, kteří je neustále používají.

FLAG – nastavuje atributy souborů

`FLAG [soubor [volby]]`

Umožňuje zobrazovat a měnit atributy **A**, **CI**, **DI**, **H**, **P**, **RI**, **Ra**, **Ro/Rw**, **S**, **Sy**, **T**, **Wa** a **X**. Dále je možno použít symboly **ALL** (všechny atributy), **N** (Normal – zruší všechny kromě Rw) a **SUB** (i pro podadresáře). Lze používat žolíkové znaky * a ?. Přidání atributu se realizuje znakem +, odebrání –.

Příklad:

Příklad přidává atribut Shareable do adresáře `SYS:PROGS` a všech jeho podadresařů všem souborům s příponou exe.

```
FLAG SYS:PROGS/*.EXE +S /SUB
```

Druhý příklad vypisuje atributy souborů v adresáři.

```
Z:\>flag
     ASK.CO_          [ Rw - A - - -- - - -- -- -- -- -- ]
     REMAP.EX_        [ Rw - A - - -- - - -- -- -- -- -- ]
     MAPDRVS.OLD      [ Rw - A - - -- - - -- -- -- -- -- ]
     NET$STAN.DAT     [ Rw - A - - -- - - -- -- -- -- -- ]
     PURGE.BAT        [ Rw - A - - -- - - -- -- -- -- -- ]
     PURGE!.EXE       [ Ro - A - - -- - - -- -- -- DI RI ]
     USERPROP.EX_     [ Rw - A - - -- - - -- -- -- -- -- ]
     LOGIN!.EXE       [ Ro - A - - -- - - -- -- -- DI RI ]
     LOGIN.BAT        [ Rw - A - - -- - - -- -- -- -- -- ]
```

FLAGDIR – nastavuje atributy adresářů

```
FLAGDIR [adresář [volby]]
```

Pracuje s atributy **D**, **H**, **N**, **P**, **R** a **Sy**. Pokud se atributy neuvedou jako parametry, vypíší se aktuální.

Příklad:

```
F:\>flagdir
SALOME/APPS:HOME/UCITELE/VAIS
VAIS Normal
```

PURGE – fyzické zrušení souborů

```
PURGE [adresář] [/ALL]
```

Pokud má server na svých discích dost místa, tak soubory smazané příkazy DEL, ERASE, různými diskovými manažery (VC, M602) nebo jinými programy

(FILER, WINDOWS) nesmaže doopravdy, ale pouze je přesune na jiné místo, kde je drží, dokud si to může dovolit. Výjimkou jsou soubory s nastaveným atributem P, ty jsou zrušeny hned. Ostatní soubory je možno obnovit použitím programu `SALVAGE`. Při použití příkazu `PURGE` dojde k fyzickému zrušení, čili takto zrušené soubory jsou již neobnovitelné. Jako parametr je možno zadat buď adresář – pak jsou smazány všechny soubory v něm, nebo pomocí žolíkových znaků * a ? specifikovat požadované soubory. Další volba `/ALL` zajistí, že se pročistí i podadresáře.

Příklad:

```
PURGE M:\DATA\*.JPG /ALL
```

MAP – mapuje disky a adresáře

```
MAP [mnoho voleb]
```

Tomuto příkazu se budeme věnovat poněkud podrobněji, protože se jedná o příkaz nesmírně užitečný. Termín „mapování" je vysvětlen v části na straně 15 – to jen pro ty, kteří neví, o čem je řeč.

Ještě než přistoupíme k vlastnímu seznámení s příkazem, dovolil bych si malou poznámku, která se bude týkat lomítek. Při práci s příkazy Novell NetWare je jedno, zda používáte k oddělování adresářů normální lomítko (/) nebo zpětné lomítko (\). Pouze v případě, že se odkazujete na nějaký soubor způsobem `\\server\svazek\adresář\podadresář\soubor`, musíte použít zpětná lomítka. Tento formát bohužel nepodporují všechny programy, proto bývá výhodnější adresář nejprve namapovat a pak se na soubor odkázat pomocí písmene disku. Tedy místo třeba `\\TYTO\SYS\PROGS\UTIL\CZECH\CSETVGA` použijte `MAP H:=TYTO/SYS:PROGS` a `H:\UTIL\CZECH\CSETVGA`.

Nyní tedy konečně k formátu příkazu:

1. `MAP [symbol:]` – bez parametrů vypíše namapování všech disků, s parametrem pouze zvoleného disku.

Příklad:

```
F:\>map

Drive  C:   maps to a local disk.
Drive  D:   maps to a local disk.
Drive  E:   maps to a local disk.
Drive  F: = COMENIUS\VOL1:HOME\SLUZBY\VAIS  \
Drive  G: = COMENIUS\SYS:PROGS  \
Drive  H: = OTUS\USR:  \
Drive  K: = TYTO\SYS:MAIL\66010001  \
Drive  M: = TYTO\US1:STUDENTS\VAIS\PUBLIC  \MYUTIL
Drive  Q: = TYTO\SYS:PROGS  \
Drive  R: = AQUILA\SYS:PROGS  \
Drive  X: = TYTO\US1:STUDENTS\VAIS  \

      -----

SEARCH1:  = Z:\ [TYTO\SYS:PUBLIC  \]
SEARCH2:  = C:\BAT
SEARCH3:  = C:\DOS
SEARCH4:  = C:\LIANE
SEARCH5:  = W:\ [TYTO\US1:STUDENTS\VAIS\BAT  \]
SEARCH6:  = V:\ [TYTO\SYS:PROGS\BAT  \]
SEARCH7:  = U:\ [TYTO\US1:STUDENTS\VAIS\PUBLIC\MYUTIL  \]
SEARCH8:  = T:\ [AQUILA\SYS:PROGS\BAT  \]
SEARCH9:  = S:\ [AQUILA\SYS:PROGS\UTIL  \]
```

```
SEARCH10:  = P:\ [COMENIUS\SYS:PROGS\BAT  \]
SEARCH11:  = O:\ [COMENIUS\SYS:PROGS\UTIL  \]
```

2. `MAP symbol:=adresář` namapuje na zvolený disk adresář ze serveru.

Příklad:

```
MAP X:=TYTO/USR:HOME/STUDENTS/VAIS
```

Adresář není kořenový, lze se tedy přepínat i do nadřazených adresářů `STUDENTS`, `HOME` atd.

3. `MAP INS[ERT] Sčíslo:=adresář` namapuje adresář jako určený k prohledávání při spouštění programů. Když se spouštěný program nenachází v aktuálním adresáři, začnou se procházet takto namapované adresáře + adresáře navolené příkazem PATH v DOSu. Číslem se zadává pořadí při procházení adresářů při hledání. `INS` zajistí, že se např. při použití *S3* nepřepíše třetí položka v proměnné PATH naší hodnotou, ale že se naše hodnota vloží před třetí položku, ze které se tak stane čtvrtá, ze čtvrté pátá atd.

Příklad:

```
MAP INS S3:=TYTO/SYS:PROGS/BAT
```

4. `MAP N[EXT] adresář` mapuje stejně jako v případě 2, písmeno disku se ale vybere automaticky, nemusí se zadávat.

5. `MAP DEL[ETE] symbol:` zruší mapování.

Příklad:

```
MAP DEL P:
```

nebo

```
MAP DEL S2:
```

6. `MAP ROOT [volby] symbol:=adresář` zajistí, že namapovaný adresář bude tvořit fiktivní kořen disku, nebude se tedy možno přepínat do vyšších úrovní adresářového stromu. `ROOT` je možno použít samostatně nebo s volbami `INS` a `N`.

Příklad:

```
MAP ROOT INS S1:=TYTO/SYS:PUBLIC
```

NCOPY – kopírování souborů na síti

`NCOPY soubor1 [TO] [adresář] [soubor2] [volby]`

Příkaz slouží ke kopírování souborů na síti. Jeho výhodou je, že je rychlejší než jeho DOSový bratr `COPY`. Vedle běžných úkonů, které známe od příkazů `COPY` a `XCOPY` z DOSu, nabízí tento program něco navíc. Pokud se kopíruje ze síťového disku na nesíťový, není vždy možno zachovat všechny atributy souboru. Pokud spustíte program s volbou `/I`, obdržíte při takovýchto stavech varování.

Program není natolik důležitý, aby si zasloužil více naší pozornosti. Zájemce opět odkazuji na manuál a nápovědu.

NDIR – vypisuje informace o souborech a adresářích

`NDIR [adresář] [soubor] [volba]`

Příkaz vypisuje jednak informace jako jeho obdoba v DOSu, příkaz `DIR`. Navíc umí pracovat s atributy NetWare a dokáže také zjišťovat majitele souborů a adresářů, čehož se hojně využívá. Opět poskytuje spoustu voleb, které jsou popsány v manuálu nebo je získáte po zadání `NDIR /?`.

Příklad:

Jako příklad si uveďme příkaz ve formátu, kdy vyhledá všechny soubory, které patří uživateli ADAMA.ZIZIEN. `/S` značí, že se mají prohledat i podadresáře, `/FO` značí, že objektem našeho zájmu jsou jen soubory.

```
NDIR X:\*.* /S /FO /OW EQ ADAMA.ZIZIEN
```

SALVAGE – obnovení smazaných souborů

```
SALVAGE
```

Jedná se opět o interaktivní program, pomocí kterého je možno obnovit již smazané soubory ze síťových disků. Celý systém funguje trošku jinak než v DOSu, kde lze obnovit při troše štěstí pouze poslední uloženou verzi smazaného souboru. Zde lze obnovovat soubory i starších verzí, po spuštění programu je k dispozici soubor jednoho jména třeba v deseti verzích, podle data a času smazání si lze vybrat soubor ve stavu, který chceme získat.

SEND – posílání zpráv

```
SEND " text" to adresát
```

Příkaz odešle text v uvozovkách adresátovi. Adresátem může být číslo spojení, jméno přihlášeného uživatele, jméno skupiny, EVERYONE (všem) nebo CONSOLE (obrazovka serveru). S uživatelskými jmény je trochu problém, pokud jsou jednoduchá (PEPA, SATRAPA, UCTARNA), jde to, pokud jsou ale složitá (VLADIMIR.VAIS), neodešlete nic. V takovém případě je lepší nejprve zjistit číslo spojení (`USERLIST VLADIMIR.VAIS`) a na to to pak odeslat (`SEND`

```
"Pojďme už domů!" TO 154).
```

Příklad:

```
Z:\>send "Nazdar" to 5
Message sent to SALOME/JAN.VLACH (station 5).
```

CASTOFF – zakáže přijímání zpráv na stanici.

```
CASTOFF [all]
```

Bez parametru jsou povoleny zprávy (SEND) pouze z konzole serveru, s parametrem zakáže příjem všech zpráv.

CASTON – povolí příjem zpráv.

```
CASTON
```

Opak příkazu CASTOFF.

NPRINT – tiskne v síti

```
NPRINT soubor [volby]
```

Jedná se o obdobu DOSového PRINT. Příkaz obsahuje podobné volby jako příkaz CAPTURE, který při použití NPRINT vůbec nepotřebujeme. Jelikož NPRINT umí v podstatě to, co CAPTURE a PRINT[5] dohromady, nebudeme se jim hlouběji zabývat.

PCONSOLE – ovládání tiskových úloh

```
PCONSOLE
```

Jedná se o interaktivní program, který se ovládá pomocí menu. Běžným uživatelům dovoluje přidávat úlohy pro tisk, mazat své úlohy, upravovat je a číst informace o cizích úlohách. Tzv. Print_Operators mohou upravovat a mazat i cizí úlohy. Smí ovládat přímo tiskárnu (odřádkovat, odstránkovat, vymazat

[5] na síti užijte místo 'PRINT' příkaz 'COPY soubor LPTx', protože COPY není rezidentní

buffer apod.) a také smí zakazovat přidávání úloh do fronty a tisk úloh, které již jsou ve frontě. Nejvíce toho smí Supervisor, může vytvářet nové fronty, definuje Print_Operators a uživatele tisku, může měnit heslo pro Print Server apod.

CAPTURE – slouží k přesměrování portů LPT stanice na síťovou tiskovou frontu.

```
CAPTURE [port s=server q=fronta nb nt nff ti=čas]
```

Uvedený formát příkazu není úplný, v praxi se ale příkaz používá právě takhle. Zájemce o detaily odkazuji na manuál. Nyní tedy k významu jednotlivých parametrů:

- **port** značí port, který se přesměrovává
- **s=server** značí server, přes který se bude tisknout
- **q=fronta** značí tiskovou frontu
- **nb** zakáže tisk stránky s informacemi o tisknuté úloze (šetří papír)
- **nt** zakáže nahrazování znaku tabelátor mezerami
- **nff** zajistí, že se po dotisknutí neodstránkuje
- **ti=čas** zajistí, že se úloha vytiskne, jakmile od posledního příchodu dat do fronty uplyne nastavený čas v sekundách. Má význam u programů, které neposílají do fronty znak EOF (konec souboru). U takovýchto programů by se s tiskem začalo až po opuštění programu (T602, IDOS). Při použití parametru se začne tisknout po uplynutí doby od ukončení posílání dat do fronty. Pozor, na pomalých počítačích nastavujte čas delší, pokud totiž program generuje data pro tisk tak pomalu, že mezery, ve kterých nic neposílá, jsou delší než nastavený čas, dochází k uzavření úlohy a jejímu vytištění.

Příklad:

```
X:\>CAPTURE LPT1 S=FOX Q=LJ5 NB NT NFF TI=30
Device LPT1: re-routed to queue LJ5 on server FOX.
```

ENDCAP – ruší platnost příkazu CAPTURE

```
ENDCAP [volby]
```

K dispozici jsou tyto volby:

- **L=číslo_portu** určuje port, který bude zlokálněn
- **ALL** pro všechny porty zruší přesměrování na síťovou tiskovou frontu
- **C** zajistí, že dosud nevytisklé úlohy budou zrušeny
- **CL=číslo_portu** je **C** a **L** pod jednou střechou
- **CALL** je **C** a **ALL** pod jednou střechou

Příklad:

```
ENDCAP L=1
```

RPRINTER – zpřístupnění lokální tiskárny pro NetWare

```
RPRINTER [pr_server číslo_tisk] [-R] [-S]
```

Pokud není tiskárna připojena přímo k serveru nebo k síťovému vedení, může se připojit k libovolné stanici v síti. Aby ji mohli používat i ostatní, musí se pomocí tohoto rezidentního programu připojit k tiskovému serveru. Program lze spustit buď s parametry nebo bez. Při spuštění bez parametrů se objeví oblíbená modrá obrazovka s výběrovým menu. Parametr -R odstraní

program z paměti, parametr -S zobrazí stav tiskárny. Číslo tiskárny lze zjistit buď v tomto programu nebo třeba v PCONSOLE.

Příklad:

```
RPRINTER AQUILA 7
```

GRANT – přiděluje práva k souborům a adresářům

`GRANT práva [adresář|soubor] TO [USER|GROUP] jméno`

Pomocí tohoto programu se přidělují prává k adresářům nebo souborům. Kromě písmen značících práva lze použít ještě slova **ALL** – všechna práva (mimo S) a **N** – žádná práva. Pomocí N je možno omezit uživateli přístup k podadresářům adresáře, ve kterém práva má. ALL má za úkol zjednodušit zadávání práv, k dispozici k ALL je ještě slovo **BUT**, které znamená „kromě".

Příklad:

Náš příklad přiděluje práva R, W, C, E, M a F do aktuálního adresáře uživateli TOMAS.RACHAC

```
Z:\>grant all but a to tomas.rachac

SALOME/SYS:PUBLIC

PUBLIC                          Rights set to [ RWCEMF ]
```

REMOVE – odebírá uživatele ze seznamu oprávněných

`REMOVE [USER|GROUP] jméno [FROM] [adresář] [/SUB]`

Při přidělení práv pomocí některého z nástrojů NW se uživatel nebo skupina stává tzv. **Trustee** (správce, důvěrník). Tato informace je uložena pouze v adresáři (nebo souboru), kde byla práva přidělena, do podadresářů se sice dědí, ale zaznamenána v nich nikde není. Pomocí příkazu lze toto Trustee zru-

šit. Při použití parametru /SUB jsou prohledány i podadresáře. Pozor, ne vždy uživatele nebo skupinu tímto způsobem omezíte, někdy to může být právě naopak.

Příklad:

Skupina ZACI má v adresáři SYS:PROGS práva R a F. Uživatel JANEK je členem skupiny a protože je šikovný, má na starosti grafické programy, má tedy práva R, W, C, E, M, F v adresáři SYS:PROGS/GRAPHICS. Ovšem adresář SYS:PROGS/GRAPHICS/PHOTOSHP mu správce nesvěřil, JANEK je zde jako Trustee jen pro čtení (GRANT R F TO JANEK). Při reorganizacích byl starý správce serveru propuštěn a nový nenechal na kameni kámen. Náhodou si všiml, že JANEK má práva R a F v adresáři PHOTOSHP, to mu přišlo zbytečné, protože věděl, že JANEK je ve skupině ZACI, která tam má také R a F. Prohodil něco o inteligenci starého správce a JANKA z adresáře PHOTOSHP odstranil. Bohužel nepostřehl, že tímto způsobem JANEK získal R, W, C, E, M, F z nadřazeného adresáře. Jak vidno, méně je někdy opravdu více.

Příklad:

```
Z:\>remove tomas.rachac
SALOME/SYS:PUBLIC
User "TOMAS.RACHAC" no longer a trustee to the specified directory.
Trustee "TOMAS.RACHAC" removed from 1 directories.
```

REVOKE – odebírá práva

`REVOKE práva [FOR] [adresář] FROM [USER|GROUP] jméno [/SUB]`

Odebírá práva z adresáře nebo souboru, kde byla přidělena (jinde to nefunguje, viz REMOVE).

Příklad:

```
Z:\>revoke e from tomas.rachac
SALOME/SYS:PUBLIC
Trustee's access rights set to [ RWC MF ]
Rights for 1 directories were changed for TOMAS.RACHAC.
```

RIGHTS – zobrazí efektivní práva

```
RIGHTS [adresář]
```

Příkaz vypočítá efektivní práva uživateli, který program spustil. Bez zadání adresáře to provede pro aktuální adresář.

Příklad:

```
X:\>rights
TYTO\US1:STUDENTS\VAIS
Your Effective Rights for this directory are [ RWCEMFA]
  * May Read from File.                    (R)
  * May Write to File.                     (W)
    May Create Subdirectories and Files.   (C)
    May Erase Directory.                   (E)
    May Modify Directory.                  (M)
    May Scan for Files.                    (F)
    May Change Access Control.             (A)
* Has no effect on directory.
    Entries in Directory May Inherit [ RWCEMFA] rights.
```

TLIST – vypíše správce

`TLIST [adresář [USER|GROUP]]`

Program vypisuje tzv. Trustees (vysvětleno u příkazu `REMOVE`). Bez parametrů program funguje pro aktuální adresář, vypisuje jak uživatele, tak skupiny. To lze omezit parametry `USER` (uživatel) a `GROUP` (skupina).

Příklad:

```
Z:\>tlist
SALOME\SYS:PUBLIC
No user trustees.
Group trustees:
  EVERYONE                              [ R    F ] Registrovaní už...
```

SYSCON – konfigurace síťového prostředí

`SYSCON`

Jedná se opět o interaktivní program s menu. Běžný uživatel si v něm může měnit svůj Login Script a heslo, ostatní informace si může pouze prohlížet. Dozví se zde spoustu informací o serveru, uživatelských skupinách, uživatelích, ale hlavně o sobě (do kdy platí účet v síti, kdy propadne heslo, kde má uživatel práva, jaké jsou kvóty na discích atd.). Program si rozhodně zaslouží vaši pozornost. Ještě lépe je na tom Supervisor, který může nastavené hodnoty měnit, může vytvářet a mazat uživatele a skupiny. Navíc má k dispozici položky, které normální uživatel vůbec nevidí: editace startovacího souboru serveru, odblokování uživatele zablokovaného detekcí vetřelců[6] atd.

FCONSOLE – umožňuje provádět ze stanice operace, které jsou jinak k dispozici jen na konzoli serveru

[6] Po několika neúspěšných pokusech o přihlášení se účet na nějaký čas zablokuje.

FCONSOLE

Běžnému uživateli je program skoro k ničemu, protože mu neumožňuje skoro nic. Uživatelé s oprávněním Console Operators mohou prostřednictvím programu měnit systémový čas, povolovat a zakazovat přihlašování k serveru a rozesílat zprávy z konzole. Nejlépe je na tom Supervisor a uživatelé mu ekvivalentní, ti mohou odpojovat jednotlivé uživatele a ukončit činnost serveru.

Část II

Protokol IPX

V této druhé části publikace se postupně seznámíme se síťovým komunikačním protokolem IPX, jeho implementací v programovacím jazyku Turbo Pascal a vším, co je ještě potřeba znát k pochopení práce IPX.

1 Referenční model OSI

Síťové komunikační služby jsou v podstatě sadou funkčních volání, která umožňují programátorovi přístup a manipulaci se síťovými protokoly IPX a SPX. Komunikace mezi jednotlivými počítači je typu **rovný s rovným**, proto nezáleží na tom, jaké stanice mezi sebou „hovoří" (server není víc než stanice, Pentium víc než 386). Aby byla komunikace mezi počítači nějak standardizována, vytvořila mezinárodní organizace ISO komunikační model, který nese jméno **OSI RM** (Open System Interconnection Referrence Model).

1.1 Vrstvy OSI modelu

Většina počítačových síti je organizována jako posloupnost vrstev (layers), kde každá z nich využívá vlastnosti té pod sebou, aniž by cokoli „věděla" o těch nad sebou nebo pod sebou. Vrstvy, které spolu komunikují, se na nazývají **partnerské vrstvy**. Vyšší vrstvy mezi sebou komunikují pouze pomocí prostředníků, pouze nejnižší vrstvy spolu komunikují v pravém slova smyslu, to jest např. po kabelu.

Pro snazší pochopení si uveďme oblíbený příklad komunikace dvou ředitelů.

	Hydroglobus, s. r. o.	Bioflor, v. o. s.
vrstva ředitel	Ing. Marcel Kapka	RNDr. Patrik Kolář
vrstva sekretariát	Mgr. Karla Mastná	Ingeborg Adamcová
vrstva podatelna	Udo Zweige	Mercedes Bukinová
vrstva doprava	Alois Volant	Kurt Satrapa
vrstva pošta	pošta Mokrá Lhota	pošta Vepříkov

Vezměme, že chce napsat ředitel Kapka řediteli Kolářovi k narozeninám. Řekne tedy své sekretářce Mastné, ať mu napíše. Ta sedne k počítači a napíše: *Milý PK,*

je to k neuvěření, jak ten čas letí. Než se nadějeme, budou z nás dědkové nad hrobem. Kde jsou ty časy, když jsme chodívávali místo deskriptivní geometrie k Beránkovi... *S pozdravem Tvůj MK*

Potom vezme pastelky, namaluje nějaký hezký obrázek a připojí k dopisu lístek s textem: *Inge, kup starýmu nějakou kytku. Díky Kadla* Vše vloží do desek, přidá cedulku „Expresně řediteli Biofloru" a předá to do podatelny, kde Udo najde v adresáři adresu Biofloru, dopis a lístek pro Adamcovou vloží do nadepsané obálky, zalepí ji a vloží do pytle s poštou, který pak pan Volant odveze na poštu, která dopis nějakým způsobem doručí do Vepříkova. Tam si na místní poště vyzvedne dopisy pan Satrapa, přiveze je slečně Bukinové, která podle rozdělovníku určí, že dopis patří na sekretariát, a tak ho donese Ingeborg Adamcové, která ho rozlepí, přečte si vzkaz, dojde koupit květinu a spolu s dopisem ji pak předá svému šéfovi.

Přitom každá (u lidí to tak asi vždy není, ale u počítačů ano) vrstva komunikuje jen s tou svou partnerskou, tzn. ředitel píše řediteli, sekretářka sekretářce atd. Možná vás napadne, že třeba poštovní vrstva by mohla mít ještě spoustu podvrstev. Máte pravdu. Stejně tomu je i při síťové komunikaci. Jelikož by ale zákonitě muselo vzniknout vrstev velké množství, udělal se kompromis. Požadovalo se, aby vrstev bylo málo a aby byly, pokud možno, co nejjednodušší.

Nakonec vzniklo vrstev celkem sedm.

1. Fyzická vrstva (physical). Tato vrstva závisí na hardwaru. Stará se o syrový přenos bitů. Definuje kabely, konektory, napětí, kódování signálu atd.

2. Spojová vrstva (data link). Zde se definuje, jak se bude logicky pracovat s médiem, tj. jak budou ošetřeny chyby, jak budou vypadat pakety (balíčky dat), jak se bude přistupovat k přenosovému médiu.

3. Síťová vrstva (network). Tato vrstva řídí práci sítě, posílá komunikační pakety, směruje (hledá cesty k cíli, vyvažuje zátěž, umí komunikovat s jinými sítěmi). Protokolem síťové vrstvy je např. IPX.

4. Transportní vrstva (transport). Tahle vrstva zaručuje, že přicházející data budu ve správném pořadí a nezdvojená. Vrstva odpovídá za rozkouskování do paketů a opětovné složení, spravuje spojení, umí rozlišovat aplikace (multiplexing) – dva počítače si na jednom portu posílají data přes WWW a na druhém přes FTP. K zástupcům patří mj. SPX.

5. Relační vrstva (session). Zde se zajišťují některé doplňkové vlastnosti jako přátelské ukončení spojení, řízení dialogu, synchronizační body, přidělování čísel symbolickým jménům stanic.

6. Prezentační vrstva (presentation). Tato vrstva se zabývá významem přenášených dat, jak prezentovat data, jak přepravovat (komprese, šifrování, kódování).

7. Aplikační vrstva (application). V této vrstvě se vytváří styk mezi sítí a libovolným aplikačním programem. Jedná se o implementaci všeobecně

užívaných služeb jako elektronická pošta (X.400), přenos souborů či adresářové služby (X.500).

2 Protokol IPX slovně

Konečně jsme se dostali k hlavnímu objektu zájmu této publikace, ke komunikačnímu protokolu IPX (Internetwork Packet eXchange). Vedle tohoto protokolu je též hodně využíván jeho kolega, protokol SPX (Sequenced Packed eXchange). Základní rozdíl je v tom, že protokol IPX se používá pro komunikace bez navázání spojení, zatímco SPX navazuje spojení. Jako příklad ze života pro IPX mějmež poštu. Každý dopis se předává samostatně. Příkladem SPX ze života je třeba telefonní hovor. Dva účastníci hovoru spolu naváží spojení, hovoří a nakonec spojení zruší.

2.1 Protokoly v NetWare

Než přejdeme k samotnému protokolu IPX, seznámíme se ještě s protokoly, které jsou používány Novell NetWare.

2.1.1 IPX

IPX (Internetwork Packet eXchange) je klíčovým protokolem NetWare a také klíčovým objektem našeho zájmu. Pro použití je jednodušší, data jsou přepravována rychleji, ale bez záruky. Protokol IPX přesně koresponduje se síťovou vrstvou podle definice v OSI RM – provádí adresování, směrování a doručování paketů k jejich adresátům. NetWare Shell používá IPX při komunikaci se serverem i přes uvedenou „nespolehlivost", protože každý požadavek od stanice vyžaduje odpověď od serveru, čímž je zajištěno, že požadovaná operace byla skutečně provedena.

2.1.2 SPX

SPX (Sequenced Packet eXchange) je na spojení orientovaný protokol, před výměnou dat pomocí SPX se navazuje spojení. Po jeho navázání je možno začít s přepravou dat. SPX zaručuje doručení *všech* odeslaných paketů ve *správném pořadí*. K přepravě SPX využívá IPX. Protokolu SPX využívá např. administrátorský nástroj RCONSOLE, který umožňuje pracovat na stanici stejně jako na konzoli serveru.

2.1.3 RIP

RIP (Routing Information Protocol) je protokol, pomocí kterého si IPX směrovače mezi sebou vyměňují směrovací informace. Z nich jsou pak vytvořeny směrovací tabulky.

2.1.4 SAP

SAP (Service Advertisement Protocol) je protokol, kterým jednotlivá zařízení v síti ohlašují (nabízejí) své služby. Každá stanice má stále čerstvé informace o tom, co je na síti k dispozici. Protokolem SAP se např. ohlašují servery, které pak zobrazí program SLIST – viz strana 35.

2.1.5 NCP

Pomocí NCP (NetWare Core Protocol) komunikují stanice se serverem. Protokol umožňuje přístup k síťovým diskům, tiskárnám a dalším sdíleným zařízením, systémové databázi atd. Pomocí tohoto protokolu se uživatelé přihlašují k serveru NetWare, také se s ním např. připojují disky NetWare k unixovým serverům.

2.2 Popis protokolu IPX

Jak již bylo řečeno v předchozím odstavci, IPX představuje v Novell NetWare nejnižší a nejrychlejší způsob komunikace, proto ho také NetWare shell používá pro vnitřní síťovou komunikaci. IPX pracuje bez navázání spojení, tzn. jakmile jedna stanice začne vysílat, musí být již adresát(-ti) připraven(-i) naslouchat. Bohužel neexistuje žádná „protokolová" možnost jak ověřit, jestli byla přijata všechna data a jestli byla přijata ve správném pořadí. Úspěšnost doručení paketu IPX je asi 95 %.

2.2.1 Adresování IPX paketů

Pro identifikaci příjemce a odesilatele paketu se používá tzv. *mezisíťová adresa*, která je složena ze tří částí: *čísla sítě*, *adresy uzlu* a *čísla soketu*.

- *číslo sítě* identifikuje jednotlivé segmenty sítě, číslo definuje správce sítě
- *adresa uzlu* identifikuje síťovou kartu, toto v celé síti jedinečné číslo bývá vypáleno v paměti karty nebo se nastavuje pomocí propojek na síťové kartě
- *soket* identifikuje aplikaci – na jednom počítači může komunikovat několik aplikací pomocí IPX, přičemž každá používá své sokety, čímž je zajištěno, že dostane pouze data, která jsou jí určena; ale i jediná aplikace může mít otevřeno více soketů a každý využívat k jinému účelu

Při použití IPX jakožto nespojované služby je možné adresovat nejen jednu stanici, ale i všechny. Vysílání pro všechny se říká *broadcasting*.

Pro odeslání paketu je nutno specifikovat celou mezisíťovou adresu, pro příjem paketu stačí pouze otevřít příslušný soket a čekat.

2.2.2 Úkony před a po přenosu dat pomocí IPX

Pokud chtějí nějaké dva počítače na síti komunikovat přes rozhraní IPX, musí aplikace na těchto počítačích otevřít soket použitím procedury *OpenSocket*. Číslo soketu musí být v obou případech shodné.

Pro komunikaci dále obě stanice potřebují znát síťové adresy svých komunikujících partnerů. Tyto adresy je možno získat pomocí funkce *GetInternetAddress*. K získání adresy je potřeba pouze znát číslo připojení (connection number) přihlášeného uživatele na požadované stanici. Toto číslo lze získat od programu USERLIST nebo, což bude možná praktičtější, pomocí procedury *GetUser* z jednotky *Novell2*, kterou najdete na přiložené disketě. Při získávání adresy si dejte pozor, když je uživatel přihlášen pod stejným jménem na serveru vícekrát. V tom případě musíte nějakým způsobem ještě zjistit, která ze získaných adres je pro vás ta pravá. Třeba požádejte adresáta, ať vám příkazem USERLIST zjistí své číslo spojení se serverem.

Po skončení komunikace již stačí jen zavřít použité sokety. K tomuto účelu slouží procedura *CloseSocket*.

2.2.3 Posílání a příjem IPX paketů

Poté, co je otevřen soket na jedné stanici a zjištěna celá mezisíťová adresa druhé stanice a naopak na druhé stanici, je již všechno připraveno k vlastní výměně dat prostřednictvím protokolu IPX s využitím procedur *SendPacket* a *ReceivePacket*.

Aby se vůbec dalo pomocí IPX komunikovat, musí každá strana připravit dvě speciální datové struktury – **Event Control Block (ECB)** a **hlavičku IPX paketu**. Struktura ECB se používá pro sledování pohybu paketů při vysílání a příjmu. Obsahuje také informace o stavu komunikace a o umístění složek paketu v paměti počítače. Detailní popis ECB je na straně 63.

Přijímání dat je ta jednodušší část komunikace. Po otevření soketu se vyplní

některé položky ECB, konkrétně musíte zadat číslo soketu, který jste otevřeli, dále zadat, kam má IPX uložit přijatá data. Přišedší paket je složen z hlavičky a za ní následujících dat, čili musíte mít pro přijetí paketu připravena dvě místa v paměti. Po dokončení události je příslušně nastavena položka ECB *Completion Code.*

K odeslání IPX paketu je zapotřebí v ECB také vyplnit některé položky. Tak samozřejmě je to číslo soketu, který byl pro tento účel otevřen. Dále musíme specifikovat data, která budou odeslána, včetně hlavičky IPX paketu. Zapotřebí je také dosadit adresu můstku, který data nasměruje k cíli – do položky *Immediate Address.* K získání této hodnoty lze využít funkci *GetLocalTarget.* Pokud je cílová stanice ve stejné síti jako vysílající, není třeba funkci volat, stačí jako hodnotu zadat její uzlovou adresu. Nejdůležitějším údajem pro doručení je adresa cílové stanice; jak bylo vysvětleno v části 2.2.2. K úspěšnému odeslání dat se ještě v ECB vyplňují další položky jako např. typ paketu apod. O těchto položkách se podrobněji zmíníme v části o ECB a v ukázkovém příkladu na posílání IPX paketů.

2.3 Struktura IPX paketu

Checksum má při použití IPX vždy hodnotu 0FFFFh. Tato položka se využívá jen z důvodu kompatibility.

Length určuje délku celého paketu (hlavička + data). Minimální délka paketu je 30 bajtů (hlavička), maximální 576 bajtů.

Transport Control je položka, kterou využívá NetWare pro konstrukci mezisíťových můstků. IPX tuto položku před posláním paketu vždy vynuluje.

Packet type označuje typ paketu. Uživatel IPX by měl vždy nastavit buď typ 0 nebo 4.

Posunutí	Název	Typ
00h	Checksum	WORD
02h	Length	WORD
04h	Transport Control	BYTE
05h	Packet Type	BYTE
06h	Destination Network	BYTE[4]
0Ah	Destination Node	BYTE[6]
10h	Destination Socket	WORD
12h	Source Network	BYTE[4]
16h	Source Node	BYTE[6]
1Ch	Source Socket	WORD
1Eh	Přenášená data	BYTE[0..546]

Tabulka 1: Struktura IPX paketu

0 Neznámý typ

1 Informační paket o cestě

2 Paket odpovědi

3 Paket indikace chyby

4 Výměnný paket

5 Řazený paket nebo paket protokolu

16–31 Pokusný protokol

17 Vnitřní protokol NetWare

Destination Network obsahuje číslo sítě, do které patří cílová stanice. Pokud jsou v této položce samé nuly, znamená to, že cílová stanice je ve stejné síti jako vysílající.

Destination Node obsahuje fyzickou adresu cílové stanice. Pokud jsou všechny bity vyplněny (FFFFFFFFFFFFh), bude paket určen všem.

Destination Socket určuje adresu soketu procesu, kterému je paket určen. Sokety vytvářejí cesty k procesům, které běží na jedné síťové adrese. Definovány jsou tyto sokety:

1	Informační paket o cestě
2	Paket odpovědi
3	Paket zpracování chyby
20h–3Fh	Pokusný paket
1–0BB8h	Sokety registrované firmou Xerox
0BB9h–	Sokety k volnému použití

Sokety pro použití v Novell NetWare:

451h	Paket obsluhující manipulaci se soubory
452h	Služební oznamovací paket (SAP)
453h	Informační paket o cestě
455h	Paket NetBiosu
456h	Diagnostický paket

Source Network obsahuje číslo sítě, ve které je zdrojová stanice.

Source Node obsahuje číslo odesilatelské stanice.

Source Socket oznamuje číslo soketu, ze kterého se vysílá.

2.4 Popis ECB

Je struktura, která obsahuje informace pro poslání a příjem paketu. ECB se obyčejně dělí na přijímací a vysílací ECB, čímž se značí jeho funkce. ECB není, narozdíl od hlavičky, nezávislý na použitém operačním systému. My zde budeme popisovat ECB pro MS-DOS.

2.5 Struktura ECB

ECB blok je tvořen dvěma částmi. Pevná má délku 36 bajtů, druhá se mění podle počtu přiřazených IPX částí.

Posunutí	Název	Typ	Uspořádání
00h	Link Address	BYTE[4]	Offset–segment
04h	ESR Address	WORD[4]	Offset–segment
08h	In Use Flag	BYTE	
09h	Completion Code	BYTE	
0Ah	Socket Number	WORD	Vyšší–nižší
0Ch	IPX Workspace	BYTE[4]	
10h	Driver Workspace	BYTE[12]	
1Ch	Immediate Address	BYTE[6]	
22h	Fragment Count	WORD	Nižší–vyšší
24h	Fragment Address 1	BYTE[4]	Offset–segment
28h	Fragment Size 1	BYTE[2]	Nižší–vyšší
2Ah	Fragment Address 2	BYTE[4]	Offset–segment
2Eh	Fragment Size 2	BYTE[2]	Nižší–vyšší

Tabulka 2: Struktura ECB

Link Address je položka, kterou IPX vlastní, když systém používá ECB. Když se ECB nepoužívá, může aplikace tuto položku změnit. Doporučuje se u všech volných ECB bloků vyplnit tuto položku aplikačním programem, aby se podle ní dal udržovat seznam volných bloků. Tuto položku nemůže programátor nijak ovlivnit přímo v ECB. V jednotce VVIPX je sice implementována, ale pouze proto, aby struktura ECB odpovídala definici, jinak s ní nepracuje jediná funkce ani procedura. Programátor si může do této položky poznamenávat např. zda se tento ECB právě používá.

ESR Address obsahuje adresu aplikací definované procedury ESR[1], kterou IPX zavolá, když dokončí operaci příjmu nebo odeslání. Není-li ESR požadováno, adresa má být vyplněna nulami.

In Use Flag obsahuje nenulovou hodnotu, když se používá ECB. Když je požadavek splněn, nastaví se na nulovou hodnotu.

Možné nenulové hodnoty:

0F8h Paket byl zařazen do fronty, kde čeká na odeslání.

0FAh Tento ECB je právě zpracováván IPX.

0FBh Událost poslání nebo příjmu byla provedena, ale ECB je dočasně umístěn ve frontě čekajících na zpracování.

0FCh Událost byla nastavena pomocí AES – asynchronní plánovač událostí – mu se jím nebudeme zabývat

0FDh Událost byla označena a IPX očekává uplynutí nastaveného časového intervalu.

Completion Code indikuje koncový stav přijímaného nebo vysílaného ECB. Tato položka nemůže mít platnou hodnotu, dokud se nevynuluje *In Use Flag.*

[1]procedura, kterou IPX volá, když nastane určitá událost

Když je ECB přiřazen pro vyslání paketu, může položka nabývat hodnoty:

00h V pořádku. Paket byl úspěšně odeslán.

0FCh Zrušeno. Požadavek poslání byl zrušen.

0FDh Chyba. Paket byl špatně utvořen. Může dojít k těmto situacím:

- Celková délka paketu je menší než 30 bajtů.
- Celková délka paketu je delší než 576 bajtů.
- První fragment je příliš malý pro hlavičku paketu.
- Nastavený počet fragmentů je nulový.

0FEh Nedoručeno. Paket nemůže být doručen.

0FFh Technická síťová chyba. Chyba hardware sítě.

Když je ECB předán IPX, aby čekal na paket, může položka nabývat těchto hodnot:

00h V pořádku. Paket byl úspěšně přijat.

0FCh Zrušeno. Požadavek příjmu byl zrušen.

0FDh Přetečení. Zjištěný paket způsobil přetečení. Paket byl přijat, ale položka *Fragment Count* je nulová nebo velikost volného místa při popisu fragmentu *Fragment Descriptor List* není tak velká, aby mohl být uschován celý přijatý paket.

0FFh Uzavřeno. Soket, který ECB používá k příjmu, není otevřen.

Když je ECB předán pro zrušení události, může položka nabývat hodnot:

00h V pořádku. Událost byla zrušena.

0FCh Chyba zrušení. Událost nemůže být zrušena.

Socket Number obsahuje číslo soketu, ke kterému je přiřazen tento ECB.

IPX Workspace je čtyřbitová položka, rezervovaná pro IPX. Nemusí se inicializovat před použitím ECB a nesmí být změněna dokud IPX používá ECB. Pokud se ECB nepoužívá, může tuto položku využít aplikace.

Driver Workspace je rezervováno pro síťový ovladač. Použití je stejné jako u IPX Workspace.

Immediate Address obsahuje adresu uzlu, do kterého (odesílaný paket) nebo z kterého (přijímaný paket) je paket poslán. Je to adresa můstku v lokální síti, jestliže paket není přijat/odeslán do/z uzlu lokální sítě. Aplikace musí tuto položku nastavit, hodnotu lze zjistit pomocí volání *GetLocalTarget*. Jestliže je cílová stanice fyzicky připojena ke stejné lokální síti, je výhodné použít rovnou její uzlovou adresu.

Fragment Count je položka, která obsahuje počet bufferů, ze kterých je vytvářen paket při posílání, nebo do kterých je paket rozdělen při příjmu. Hodnota musí být větší než nula.

Fragment Descriptor označuje zdroj nebo cíl skutečné oblasti paketu, která je posílána. IPX musí mít nejméně jednu položku Fragment Descriptor a libovolný počet přídavných deskriptorů, které vytvářejí *Fragment Descriptor List*.

Fragment Descriptor je složen ze dvou položek následujícího významu:

Address	Obsahuje adresu bufferu, ze kterého je IPX paket poslán nebo do kterého je přijat.
Fragment size	Obsahuje délku bufferu, na který ukazuje Address.

Buffer fragmentu určený pro první položku v položce Fragment Descriptor List musí být minimálně 30 bajtů dlouhý, musí obsahovat kompletní hlavičku paketu IPX. Celková délka paketu nesmí přesáhnout 576 bajtů.

2.6 Procedura ESR

Event Service Routine je procedura, kterou IPX volá, když nastane zadaná událost. Událostí může být třeba splnění požadavku poslání nebo dokončení požadavku na příjem.

Procedury ESR jsou vždy volány v závěru událostí, kdy je položka *In Use Flag* vynulována, položka *Completion Code* je nastavena a všechny vhodné události, které se vztahují k události, jsou přeneseny do ECB. ECB ukazuje na ESR, čímž je umožněno její zavolání v okamžiku výskytu očekávané události.

Vzhledem k tomu, že procedura ESR je volána v režimu zakázaného přerušení procesoru, měla by být co nejrychlejší, aby se minimalizoval čas, kdy je procesor ve stavu zakázaného přerušení. Není tedy vhodné pomocí této procedury vykonávat operace, které by stejně dobře zvládl sám aplikační program.

Procedura vyžaduje při zavolání toto:

- v registru AL je hodnota 0FFh – volána z IPX
- ukazatel na ECB je v registrech ES:SI[2]
- stav všech registrů a příznaků procesoru s výjimkou SS a SP je uložen na uživatelský zásobník, ESR si uloží hodnoty SS a SP a na konci je vrátí do stavu, v jakém byly při spuštění
- zakázáno přerušení

V okamžiku zavolání ESR je položka ECB *In Use Flag* vynulována, všechny potřebné informace jsou uloženy do ECB. ESR může s ECB volně manipulovat.

- než nějaká procedura začne modifikovat systémové proměnné aplikace, musí být nastaven v ESR registr DS
- ESR nevrací žádné příznaky ani hodnoty

[2]neděste se, v našich programech budeme IPX používat poněkud lidštěji pomocí jednotky VVIPX

- návrat z ESR musí být proveden při zakázaném přerušení
- procedura ESR musí provést návrat vzdálenou instrukcí RETF, protože je volána vzdálenou instrukcí CALL

Jelikož práce s ESR vyžaduje hlubší znalosti systémového programování, nebudeme si jí více zabývat. Pokud potřebné znalosti máte, nebude pro vás problém ESR využívat, protože vše potřebné bylo řečeno.

3 Protokol IPX programově

Jak již bylo přislíbeno, nebudeme programovat v JSA. Pro naše příklady použijeme programovací jazyk Borland Pascal 7.0, protože tento jazyk je velmi dobře srozumitelný a tudíž pro výuku mimořádně vhodný.

3.1 Jednotka VVIPX

Základem našeho programování bude jednotka **VVIPX**, což je upravená verze jednotky z knihy [1].

3.1.1 Veřejné typy a proměnné

Jednotka používá několik vlastních typů proměnných:

- `NetwID = array[1..4] of byte;`
  – identifikační číslo sítě
- `NodeID = array[1..6] of byte;`
  – identifikační číslo uzlu v síti
- `IPXAddress = record`
  `Network: NetwId;`
  `Node: Nodeid;`

```
Socket: Word;
end;
```

– celá 12-bajtová adresa uzlu

- IPXhead = record

```
CheckSum: array[1..2] of byte;
Length: word;
TransCtrl: byte;
PacketTyp: byte;
DestNetw: Netwid;
DestNode: NodeId;
DestSckt: word;
SrceNetw: Netwid;
SrceNode: NodeId;
SrceSckt: word;
end;
```

– hlavička IPX paketu

- ECBblock = record

```
Link: array[1..2] of integer;
EsrAddr: pointer;
InUse: byte;
ComplCode: byte;
ScktNum: array[1..2] of byte;
IPX_w_spc: array[1..4] of byte;
Driver_sp: array[1..12] of byte;
ImmdAddr: NodeId;
FragCnt: array[1..2] of byte;
FragAdr: pointer;
FragSize: word;
```

`end;`

– struktura ECB bloku

- `NetError: byte;`

  – tato proměnná se nastavuje při každém sítovém volání, je v ní uložen kód chyby poslední síťové operace, pokud není IPX nainstalováno, dostane na začátku hodnotu 0Fh a všechny žádosti o síťové operace jsou ignorovány

- `OpenStyle: byte;`

  – určuje v proceduře *IPXOpenSocket* způsob otevření soketu, implicitně 0 pro přechodné sokety (ShortLive), programátor může změnit na 1 pro trvalé sokety (LongLive)

3.1.2 Procedury

IPXInitialize – detekuje ovladač IPX a inicializuje jednotku

```
Procedure IPXInitialize;
```

Procedura je volána automaticky po spuštění programu. Zjišťuje, je-li rozhraní IPX dostupné (proměnná *NetError* = 0).

IPXOpenSocket – otevře zadaný soket

```
Procedure IPXOpenSocket(var number: word);
```

Tato procedura se používá k otevření soketu, což je nezbytně nutné pro pozdější používání soketu.

V položce *number* zadáváme číslo požadovaného soketu. Při použití hodnoty 0000 je vybrán automaticky volný soket z intervalu 4000h a 5000h, jehož číslo bude navráceno v této položce.

Pokud je proměnná *NetError* nastavena na hodnotu 0, znamená to, že došlo k úspěšnému otevření soketu, je-li hodnota 240 (F0h), IPX není nainstalováno. Hodnota 254 (FEh) říká, že tabulka soketů je plná a hodnota 255 (FFh)

znamená, že soket již byl otevřen.

Životnost soketu je dána proměnnou *OpenStyle* – viz str. 70.

Maximální počet otevřených IPX soketů je 150 a nastavuje se v souboru SHELL.CFG, implicitní hodnota je 20. V případě, že vám to nevyhovuje, kontaktujte správce vaší sítě. Rozhodně byste neměli zasahovat do konfigurace bez jeho vědomí.

OpenSocket – otevře zadaný soket

```
Procedure OpenSocket(number: word);
```

Procedura funguje jako procedura *IPXOpenSocket* až na to, že neumožňuje otevření volně přidělitelných soketů.

GetLocalTarget – předává do IPX adresu cíl. uzlu

```
Procedure GetLocalTarget(networkAddress: IPXAddress; var
immAddress: NodeID; var TransportTime: word);
```

Vstupem procedury je položka *networkAddress*, výstupem ostatní položky a proměnná *NetError* (0 – v pořádku, 250 – nezjištěn lokální cíl paketu).

Procedura předá do IPX mezisíťovou adresu cílového uzlu. Vrací odhadnutou přenosovou dobu pro 576-bajtový paket a dále vrací adresu cílového uzlu nebo můstku, který bude použit pro doručení.

Položka *TransportTime* obsahuje odhadnutou dobu doručení v jednotkách o velikosti cca 1/18 sekundy. Hodnota je pouze orientační, může se pochopitelně změnit v závislosti na zatížení sítě a délce paketu.

Při přijmutí paketu IPX vyplní položku *immAddress* adresou vysílacího uzlu, tj. buď přímo adresou odesilatele (leží-li zdroj i cíl ve stejné síti) nebo adresou mezisíťového můstku (je-li odesilatel odjinud).

Jakmile se jednou začne s výměnou paketů s jiným uzlem, není pak již potřeba stále proceduru volat, adresu lze získat z položky *Immediate Address* přijímaného ECB. ECB, kterým byl paket přijat, lze poslat jiná data odesilateli

bez nutnosti měnit položku *Immediate Address.*

GetMyNetAddr – vrací adresu uzlu

```
Procedure GetMyNetAddr(var NodeNum: NodeID; var NetwNum:
NetwID);
```

Procedura vrací adresu uzlu, na kterém byla spuštěna. V položce *NodeNum* je adresa uzlu o délce 6 bajtů, v *NetwNum* je 4-bajtová adresa sítě.

Procedura se používá, pokud např. chceme, aby aplikace šla spustit jen na některých uzlech v síti apod.

GetInternetAddress – vrací adresu uzlu a sítě

```
GetInternetAddress(ConnectionNumber: Byte; var NetworkNumber:
NetwID; var PhysicalNodeAddress: NodeID);
```

Procedura podle zadaného čísla spojení se serverem vrací číslo sítě a uzlovou adresu stanice. Získané hodnoty se mj. doplňují do hlavičky paketu. Pokud si zkusíte porovnat získané hodnoty s hodnotami programu USERLIST, musíte vzít v úvahu, že USERLIST vypisuje čísla v šestnáctkové soustavě. U uzlových adres to je většinou patrné na první pohled, u čísla sítě už ne.

SendPacket – odešle paket

```
Procedure SendPacket(var ECB:ECBlock);
```

Procedura předá ECB síťovému driveru k odeslání k cíli.

Před zavoláním procedury je zapotřebí, aby byly vyplněny ECB-položky *ESR Address*, *Socket Number*, *Immediate Address*, *Fragment Count* a popis každého fragmentu (adresa, délka). Dále popis paketu *Packet Type* a cílová adresa. Položka *Immediate Address* může být nastavena pomocí procedury *GetLocalTarget.*

Po zavolání procedury je položka *In Use Flag* nastavena na hodnotu FFh, po odeslání je položka vynulována a položka *Completion Code* obsahuje

vhodnou hodnotu (viz dále). Nakonec se zavolá (je-li definována) ESR popsaná ukazatelem *ESRAddr*.

V *Completion Code* může být:

0 v pořádku

252 požadavek zrušen

253 špatný paket (delší než 576 bajtů, hlavička nemá 30 bajtů nebo není v prvním fragmentu)

254 paket nelze doručit (cíl neexistuje nebo není k dispozici můstek)

255 technická závada

Pozor! Že byl paket úspěšně odeslán ještě neznamená, že byl úspěšně přijat. Když vhodíte dopis do schránky, nemáte také žádnou jistotu, že ho adresát skutečně dostal. Můžete se s ním ale dohodnout, že vám pošle potvrzení, že dopis dostal. Je otázkou, zda se tentokrát někde cestou neztratí to potvrzení.

Při posílání paketu do stejné sítě, ve které stanice leží, musí být hodnota *Immediate Address* nastavena na 0FFh.

ReceivePacket – připraví IPX k přijetí paketu

```
Procedure ReceivePacket(var ECB: ECBblock);
```

Procedura předá IPX adresu ECB pro přijetí paketu. Před zavoláním musí být otevřen požadovaný soket (na rozdíl od vysílání, kde není otevření soketu nutné) a nastaveny položky ECB *ESR Address*, *Socket Number*, *Immediate Address*, *Fragment Count* a položky, popisující jednotlivé fragmenty. Pokud nebude potřeba volat ESR, musí ukazatel obsahovat hodnotu **nil** – nic.

Položka *In Use Flag* je nejprve nastavena na hodnotu 0FEh, což značí, že ECB čeká na přijetí paketu. Tento ECB je také přidán do seznamu přijímajících ECB.

Když dorazí paket, je použit jeden z čekajících ECB pro přijetí, ostatní ECB čekají dál, a to i v případě, že čekaly na stejný paket. Je vyplněna položka *Completion Code* a *Immediate Address* (vysílač nebo můstek).

V poslední fázi je nastavena položka *In Use Flag* na hodnotu 00h a případně se zavolá ESR, pokud je v *ESR Address* požadována.

Hodnoty *Completion Code* jsou tyto možné:

0	v pořádku
252	požadavek zrušen
253	špatný paket
255	soket nebyl otevřen

CancelEvent – zruší událost

```
Procedure CancelEvent(var ECB: ECBblock);
```

Procedura zruší událost IPX definovanou pomocí ECB předáním adresy ECB ovladači IPX. Návratová hodnota se ukládá do globální proměnné *NetError*, která může nabývat hodnoty:

0	v pořádku
249	událost v ECB nelze zrušit
252	událost zrušena
255	ECB se již nepoužívá

Po zrušení události již IPX nevolá proceduru ESR, popsanou v položce *ESRAddress* v ECB, a také vynuluje položku *InUseFlag*.

CloseSocket – uzavře daný soket

```
Procedure CloseSocket(number: word);
```

Procedura uzavírá daný IPX soket a ruší všechny události definované pomocí ECB k tomuto soketu. V položce *Completion Code* každého takového ECB je hodnota 0FCh a položka *InUseFlag* je vynulována.

Uzavřít lze libovolný (i neotevřený) soket. *Short Live* sokety se uzavřou po ukončení programu samy, sokety *Long Live* musí zavřít program.

Procedura nesmí být volána pomocí ESR.

3.2 Ukázkové příklady

V této části se seznámíme s praktickým použitím IPX v jazyce Turbo Pascal 7.0. První tři příklady jsou tím nejjednodušším: zjištěním adres a odesláním a přijmutím dat prostřednictvím IPX. Další příklad je praktická aplikace, která využívá právě první tři příklady. Tak to bude pravděpodobně i ve vašich vlastních programech. K dispozici budete mít navíc ještě několik jednotek z Internetu. Podrobněji v části 3.3 na stránce 90.

3.2.1 Adresy

První příklad demonstruje práci s adresami použitými v jednotce VVIPX. Nejprve jsou pomocí procedury *GetMyNetAddr* zjištěna čísla uzlu a sítě, kde byl program spuštěn. V druhé části se zjišťují pomocí procedury *GetInternetAddress* tyto adresy pro stanici s číslem připojení 5. Při testování programu se nejprve podívejte programem USERLIST, zda má některá stanice číslo připojení 5 a pokud ne, změňte hodnotu konstanty v programu na nějakou jinou podle USERLISTu. Ve třetí části zjišťujeme procedurou *GetLocalTarget* adresu prostředníka (pokud leží cíl ve stejné síti, bude to adresát, jinak můstek) a časovou náročnost doručení paketu v jednotkách 1/18 sekundy, což hned přepočítáme a vypíšeme. Znovu upozorňuji, že program USERLIST vypisuje čísla v šestnáctkové soustavě, kdežto náš program v desítkové.

Ukázka činnosti programu:

```
Tato stanice:
Uzlová adresa: 0:192:108:36:84:130
Adresa sítě: 0:0:0:18
Stanice se spojením č. 5:
Uzlová adresa: 0:192:108:36:6:96
Adresa sítě: 0:0:0:18
Adresa prostředníka: 0:192:108:36:6:96
```

```
Časová náročnost doručení: 0.11 sekund

F:\>userlist /a
User Information for Server SALOME
Connection  User Name        Network    Node Address
----------  --------------  --------   ------------
     5      JIRI.TICHY      [      12] [  C06C240660]
    10    * VLADIMIR.VAIS   [      12] [  C06C245482]
```

uses *vvipx*;

const *connection* = 5; {číslo spojení zkoumané stanice – viz Userlist}

var *Net_addr*: *IPXAddress*; {síťová adresa}

Imm_addr: *NodeID*; {adresa prostředníka}

Netw: *NetwID*; {adresa sítě}

Trans_time: *word*; {odhadnutá doba přenosu}

Node: *NodeID*; {adresa uzlu}

i: *integer*; {řídící proměnná cyklů}

begin

writeln('----------');

GetMyNetAddr(*Node*, *Netw*); {zjisti mou adresu uzlu a sítě}

writeln('Tato stanice:');

write('Uzlová adresa: ');

for *i* := 1 **to** 5 **do** *write*(*Node*[*i*], ':'); {vypiš adr. uzlu}

writeln(*Node*[6]);

write('Adresa sítě: ');

for *i* := 1 **to** 3 **do** *write*(*Netw*[*i*], ':'); {vypiš adr. sítě}

writeln(*Netw*[4]);

writeln;

GetInternetAddress(*Connection*, *Netw*, *Node*); {zjisti adresy cíle}

```
writeln('Stanice se spojením č. ', connection, ':');
write('Uzlová adresa: ');
for i := 1 to 5 do write(Node[i], ':');    {vypiš uzlovou adr. cíle}
writeln(Node[6]);
write('Adresa sítě: ');
for i := 1 to 3 do write(Netw[i], ':');    {vypiš adr. sítě cíle}
writeln(Netw[4]);
writeln;
Net_Addr.Node := Node;    {vyplň záznam Net_addr získanými hodnotami}
Net_Addr.Network := Netw;    {položku Soket nepotřebujeme}
GetLocalTarget(Net_Addr, Imm_Addr, Trans_time);    {zjisti prostředníka a čas}
write('Adresa prostředníka: ');
for i := 1 to 5 do write(Node[i], ':');    {vypiš uzl. adr. prostředníka}
writeln(Node[6]);
writeln('Časová náročnost doručení: ', Trans_time/18:2:2, ' sekund');    {a čas}
end.
```

3.2.2 Vysílač

Ukázkový program odesílá zadaný text prostřednictvím zvoleného soketu. Přijímačů může být libovolné množství, protože, jak již bylo řečeno, IPX je služba nespojovaná.

Jednotlivé řádky programu jsou očíslovány pro snadnou orientaci při rozboru programu.

```
1  Program Vysilac;
2  uses crt, dos, vvipx;
3  const socket = $5555;
4  type
5  IPXblock = record
```

```
6                 Head: IPXhead;
7                 DataBulk: array[1..538] of byte;
8               end;
9  var
10  IPXs: IPXblock;
11  ECBs: ECBblock;
12 Procedure Send_Packet;
13 var x: integer;
14      a: string;
15      delka: word;
16 begin
17  FillChar(IPXs, SizeOf(IPXs), 0);
18  FillChar(ECBs, SizeOf(ECBs), 0);
19  with IPXs.Head do
20    begin
21      FillChar(CheckSum, 2, 0);
22      TransCtrl := 0;
23      PacketTyp := 0;
24      DestNetw := unknown;
25      DestNode := vsem;
26      word(DestSckt) := socket;
27    end;
28  write('Zadejte text k odeslání: ');
29  readln(a);
30  for x := 1 to length(a) do IPXs.databulk[x] := ord(a[x]);
31  delka := 30 + length(a);
32  with ECBs do
33        begin
```

```
34          Esraddr := nil;
35          word(ScktNum) := socket;
36          ImmdAddr := vsem;
37          FragCnt[1] := 1;
38          FragCnt[2] := 0;
39          FragAdr := addr(IPXs);
40          Fragsize := delka;
41        end;
42    OpenSocket(socket);
43    SendPacket(ECBs);
44    while ECBs.InUse <> 0 do;
45    CloseSocket(socket);
46  end;   {Send_Packet}
47  begin   {main}
48    Clrscr;
49    IPXInitialize;
50    if NetError <> 0 then
51                        begin
52                          writeln('IPX není nainstalováno!');
53                          halt;
54                        end;
55    Send_Packet;
56  end.
```

Rozbor programu:

Tento program ukazuje nejjednodušší způsob, jak prostřednictvím IPX odeslat paket. Ve svých prvních programech s využitím IPX budete ve většině případů odesílat data právě tímto způsobem.

Na řádku 3 jsme si zvolili číslo soketu pro naše pokusy. Na 5. řádku je

definice záznamu *IPXblock*, který obsahuje dvě položky – *Head*, což je hlavička paketu a *DataBulk*, která bude přenášet vlastní data. Na řádcích 10 a 11 definujeme proměnné *IPXs* a *ECBs*, pomocí kterých předáme data ovladači IPX k transportu. Na 12. řádku je klíčová procedura celého programu – procedura *Send_Packet*. Před vyplňováním položek záznamů IPXs a ECBs je nejprve na řádcích 17 a 18 vyčistíme. Na 19. řádku vyplníme hlavičku. *CheckSum* se využívá jen z důvodu kompatibility, jak již bylo řečeno. *TransCtrl* (Transport Control) se musí vždy vynulovat – viz Struktura IPX paketu. Do položky *DestNetw* (Destination Network) uložíme pole typu *NetwID*, které je vyplněné nulami – cílová stanice je ve stejné síti jako ta naše vysílající. Budeme vysílat broadcast pro všechny, tudíž položku *DestNode* (Destination Node) vyplníme bitově samými jedničkami, k čemuž nám napomůže konstanta *vsem* typu *NodeID*. Nakonec na řádku 26 doplníme číslo soketu, který jsme si definovali na začátku programu. Na 29. řádku do proměnné *a* načteme textový řetězec, který budeme posílat a na řádku 30 ho vložíme do „nákladního prostoru" *IPXs.DataBulk*. Délka paketu tedy bude 30 (hlavička) + délka řetězce. Nyní ještě potřebujeme vyplnit ECB blok. ESR nechceme využívat, proto do položky *ESRAddr* vložíme hodnotu *nil*. Vyplníme číslo soketu a Immediate Address – pro všechny stanice. Do ECB položky *Fragment Count* zapíšeme, že naše data jedou vcelku, tedy jeden fragment. *Fragment Count* je typu word, tedy do nižšího bajtu zapíšeme hodnotu *1*, do vyššího *0*, jak je tomu na řádcích 37 a 38. Do *FragAddr* přiřadíme ukazatel na naše data a do *FragSize* jejich velikost. Na 42. řádku otevřeme soket a na dalším řádku předáme náš ECB blok ke zpracování ovladači IPX a čekáme, dokud není hotovo. A jakmile se tak stane, uzavřeme soket, který jsme použili, a skončíme. Hlavní tělo programu, začínající na řádku 47 má za úkol pouze zinicializovat jednotku a otestovat připravenost rozhraní IPX plnit naše požadavky.

3.2.3 Přijímač

Tento program demonstruje příjem dat prostřednictvím služeb IPX. Na všech stanicích ve stejné síti, kde tento program poběží, zachytíte data vysílaná programem *Vysílač*.

```
1  Program Prijimac;
2  uses crt, vvipx;
3  const socket = $5555;
4  type
5  IPXblock = record
6                Head: IPXhead;
7                DataBulk: array[1..538] of byte;
8             end;
9  var
10  x: integer;
11  k: char;
12  ipxs, ipxr: IPXblock;
13  ecbs, ecbr: ECBblock;
14 Procedure Rcv_Packet;
15 var x: integer;
16 begin
17 with ECBr do
18       begin
19         Esraddr := nil;
20         word(ScktNum) := socket;
21         ImmdAddr := vsem;
22         FragCnt[1] := 1;
23         FragCnt[2] := 0;
24         FragAdr := addr(IPXr);
```

```
25        word(FragSize) := 576;
26      end;
27  OpenSocket(socket);
28  if NetError <> 0 then
29                        begin
30                          writeln('Soket pro příjem nelze otevřít.');
31                          halt;
32                        end;
33  ReceivePacket(ECBr);
34  repeat until (ECBr.InUse = 0) or keypressed;
35  with IPXr.head do length := hi(length) + 256 * lo(length);
36  If ECBr.InUse = 0 then
37                        begin
38                          writeln(#10#13'Přijal jsem zprávu:');
39                          for x:= 1 to IPXr.head.length - 30 do
40                            write(chr(IPXr.Databulk[x]))
41                        end
42                      else CancelEvent(ECBr);
43  CloseSocket(socket);
44  end;   {Rcv_packet}
45  begin   {main}
46   IPXInitialize;
47   if NetError <> 0 then
48                        begin
49                          writeln('IPX není nainstalován!');
50                          halt;
51                        end;
52   clrscr;
```

```
53  writeln('Přijímám zprávy');
54  while not(keypressed) do rcv_Packet;
55  while keypressed do K := readkey;
56 end.
```

Rozbor programu:

Začátek programu opět jako předcházející příklad definuje číslo použitého soketu a *IPXblock*. Analogicky k proměnným *IPXs* a *ECBs* nyní definujeme na řádcích 12 a 13 proměnné *IPXr* a *ECBr*. Na 14. řádku programu začíná hlavní procedura *Rcv_Packet*. Stejně jako v předchozím příkladu vyplníme strukturu ECB, pouze do *FragSize* přiřadíme hodnotu *576*, což je maximální možná délka IPX paketu. Na 27. řádku otevřeme soket a na 33. předáme ECB ovladači IPX, aby nám naplnil *DataBulk* daty, až přijdou – ukazatel na *DataBulk* jsme ovladači předali v položce *Fragment Address* záznamu ECB. Na 34. řádku musíme počkat, až data dorazí (IPX vynuluje *In Use Flag*) nebo dokud někdo nestiskne klávesu. Na 35. řádku prohodíme vyšší a nižší bajt v *IPXr.Head.Length*. Tohle je prostě fakt, takhle se to dělá. Pokud nevěříte, nechte si hodnotu vypsat, hned po přijmutí paketu a pak po prohození bajtů. IPX patrně během přenosu bajty prohazuje. Na 36. řádku testujeme hodnotu *In Use Flag* pro případ, že žádná data nepřišla a čekací smyčka na řádku 34 byla opuštěna stiskem klávesy. Pokud data přišla, vyextrahujeme je z paketu a vypíšeme pomocí cyklu na řádkách 39 a 40. Pokud žádná data nepřišla (byla stisknuta klávesa), tak na 42. řádku odebereme ECB z fronty čekajících a zavřeme soket. Malá poznámka k přenášeným datům: Data přenášíme binárně, proto jsme museli textové znaky pomocí funkce *ord* konvertovat na čísla a pak zase pomocí funkce *chr* na znaky. Při přenášení binárních dat, tyto operace pochopitelně provádět nebudeme. Nyní již zbývá pouze zavřít po sobě soket, učiníme tak na 43. řádku. Hlavní tělo programu se chová téměř stejně jako v minulém případě. Otestuje připravenost rozhraní IPX a zavolá proceduru

Rcv_Packet.

3.2.4 Chat

Program *Chat* je program, který rozesílá vše, co pisatel napíše. Zprávu zachytí každý, kdo naslouchá na správném soketu, což je i samotný program, tudíž všichni, kdož si tento program pustí, spolu mohou „hovořit".

Program využívá téměř nezměněné procedury na odesílání a přijímání paketů z předcházejících dvou příkladů. Tyto procedury proto nejsou opatřeny komentáři, neboť byly již podrobně rozebrány. Zbylá část programu obsahuje komentáře přímo ve zdrojovém textu, jak je v Turbo Pascalu zvykem.

```
Program Chat;
uses vvipx, dos, crt;
const socket = $5555;
type
IPXblock = record
             Head: IPXhead;
             DataBulk: array[1..538] of byte;
           end;
var
 ipxs, ipxr: IPXblock;
 ecbs, ecbr: ECBblock;
 chat_id: string[10];    {jméno v diskusní místnosti}
 prijato: string;    {po IPX přišedší text}
 pozice: byte;    {pozice v editovaném řetězi při zadávání textu}
 radek: byte;    {na kterém zrovna řádku píšeme zprávy, co přišly}
 i, edit_x: byte;    {pomocné proměnné}
 retezec: string[80];    {editovaný text k odeslání}
Procedure Send_Packet(a: string);    {odešle po IPX řetězec A}
```

```
var x: integer;
    delka: word;
begin
 FillChar(IPXs, SizeOf(IPXs), 0);
 FillChar(ECBs, SizeOf(ECBs), 0);
 with IPXs.Head do
   begin
    FillChar(CheckSum, 2, 0);
    TransCtrl := 0;
    PacketTyp := 17;    {unknown}
    DestNetw := unknown;
    DestNode := vsem;
    word(DestSckt) := socket;
   end;
 for x := 1 to length(a) do IPXs.databulk[x] := ord(a[x]);
 delka := 30 + length(A);
 with ECBs do
       begin
        Esraddr := nil;
        word(ScktNum) := socket;
        ImmdAddr := vsem;
        FragCnt[1] := 1;
        FragCnt[2] := 0;
        FragAdr := addr(IPXs);
        Fragsize := delka;
       end;
 SendPacket(ECBs);
 while ECBs.InUse <> 0 do;
```

```
end; {Send_Packet}
procedure zpracovani; {obsluha editace textové řádky}
var key: char; {načtený znak}
    i: integer;
begin
key := readkey;
 if key = #0 then {funkční klávesy, šipky atd. začínají nulou -
  nezajímají nás}
  begin
   readkey; {přečteme ještě druhý znak}
   exit; {a zahodíme ho}
  end;
 if key in [#32..#255] then {vida, to jsou normální znaky}
   if pozice < (77 - length(chat_id)) then {řádek ještě není plný}
    begin
     gotoxy(length(chat_id) + 3 + pozice, 25);
     write(key); {tak ho napíšeme na obrazovku}
     inc(pozice); {šoupneme si pozici}
     retezec := retezec + key; {a přidáme znak do řetězce k odeslání}
     exit;
    end;
 if key = #8 then {Backspace}
  if pozice > 0 then {je co smazat}
  begin
   dec(pozice); {snížíme pozici}
   retezec := copy(retezec, 1 , length(retezec) - 1);
      {umázneme 1 znak z řetězce}
   gotoxy(length(chat_id) + 3 + pozice, 25);
```

```
    write(' ');    {a na obrazovce}
    gotoxy(wherex - 1, 25);    {a poskočíme kurzorem na jeho místo}
    exit;
   end;
  if key = #13 then    {Enter - budeme posílat}
   begin
    send_packet(chat_id + ': ' + retezec);    {pošli id + řetězec}
    retezec := '';    {vyčisti řetězec}
    pozice := 0;    {nic v něm není}
    gotoxy(1 , 25);    {vyčisti editační řádku}
    for i := 1 to 79 do write(' ');    {mezerami}
    gotoxy(1 , 25);
    write(chat_id + ': ');    {a napiš tam naše id}
    exit;
   end;
  if key = #27 then    {Esc - končíme}
   begin
    send_packet('*** Odpojil se ' + chat_id + ' ***');
      {oznámíme, že to balíme}
    writeln;
    writeln('Konec');
    CloseSocket(socket);    {zavřeme soket}
    halt;    {skončíme}
   end;
end;
Function Rcv_Packet: string;    {vrací text z paketu, který přijme}
var x: integer;
    pom, klavesa: string;
```

```
begin
pom :='';
with ECBr do
      begin
        Esraddr := nil;
        word(SocktNum) := socket;
        ImmdAddr := vsem;
        FragCnt[1] := 1;
        FragCnt[2] := 0;
        FragAdr := addr(IPXr);
        word(FragSize) := 576;
      end;
ReceivePacket(ECBr);
repeat
 if keypressed then zpracovani;    {pokud někdo něco stisknul,
  tak to obsloužíme, jinak čekáme na paket}
until (ECBr.InUse = 0);
with IPXr.head do length := hi(length) + 256 * lo(length);
If ECBr.InUse = 0 then
                    begin
                      for x:= 1 to IPXr.head.length - 30 do
                        pom := pom + chr(IPXr.Databulk[x]);
                    end
                  else CancelEvent(ECBr);
Rcv_packet := pom;    {přiřadíme data do jména funkce}
end;   {Rcv_packet}
begin   {main}
radek := 0;    {začneme na horním okraji obrazovky}
```

```
clrscr;
writeln('IPX Chat V1.0');
writeln('Vladimír Vais, 31.3.1998');
writeln;
write('Zadej své chatové jméno: .....');   {max. 10 znaků}
gotoxy(wherex - 10, wherey);
readln(chat_id);   {načti id}
if chat_id='' then halt;   {anonymy nebereme}
IPXInitialize;   {zinicializuj jednotku}
if NetError <> 0 then
                    begin
                     writeln('IPX není nainstalován!');
                     halt;   {smůla, IPX neběží}
                    end;
OpenSocket(socket);   {otevřeme soket}
send_packet('*** Připojil se ' + chat_id + ' ***');   {ohlásíme se ostatním}
pozice := 0;   {všechno vyčistíme a dáme se do toho}
retezec := '';
clrscr;
gotoxy(1, 24);
for i := 1 to 80 do write('-');   {namalujeme oddělovací čáru}
gotoxy(1, 25);
write(chat_id + ': ');   {a zkraje napíšeme naše id}
repeat
 prijato := rcv_packet;   {něco přišlo}
 if radek = 23 then radek :=1 else inc(radek);   {koukáme, abychom}
 edit_x := wherex;   {poznamenáme si pozici x v edit. ř.}   {nepsali někam mi
 gotoxy(1, radek);   {vyhrazenou oblast}
```

```
    for i := 1 to 80 do write(' ');   {vymažeme starý text}
    gotoxy(1, radek);
    writeln(prijato);   {a napíšeme, co přišlo}
    if radek < 23 then for i := 1 to 80 do write(' ');   {a smažeme řádek pro
                                                          příští zprávu}
    gotoxy(edit_x, 25);   {šup dolů do edit. řádku}
  until false;   {a tak pořád dokola}
  end.   {sem se to nikdy nedostane, ale end tu musí být :-)}
```

Pokuste se program upravit, aby bylo možné používat více diskusních „místností“ (každá na jiném soketu). Doplňte do programu službu, která zobrazí všechny účastníky (např. zasláním speciální sekvence, na kterou se každý program ohlásí).

3.3 Užitečné jednotky z Internetu

```
IPX        PAS        9,540
BINDERY2   PAS       31,845
STRINGS    PAS        3,236
BINDERY    PAS       40,277
DIRECT     PAS       15,871
ACCESS     PAS          463
ESR2INT    PAS        6,561
IPX2       PAS        7,463
TYPES      PAS        5,710
IPX1       PAS        6,482
SPKT       PAS        3,355
PKTPAS     PAS        4,918
MYBINDER   PAS        3,935
NETDOS     PAS        3,036
```

```
NEWSTAT  PAS        3,906
NOVELL2  PAS       61,226
NETWARE  PAS       50,376
PKASCIIZ PAS          755
PKINLINE PAS          936
PKSTRING PAS        3,462
```

Uvedený seznam informuje o jednotkách, které najdete na přiložené disketě. Jedná se o soubory, které jsem nasbíral během studia na VŠ a které se mi zdály z nějakého důvodu užitečné. Některé obsahují informace o činnosti v sobě, jiné budete muset prozkoumat a odzkoušet, jak pracují. Já se zde zmíním pouze o jediné, a tou jest *Novell2*. Tuto jednotku zaručeně na Internetu ani nikde jinde nenajdete, najdete ale velmi podobnou se jménem *Novell*, což byla má první novellovská jednotka, kterou jsem použil ve svém prvním síťovém programu. Jak se časem ukázalo, měla chyby, které jsem opravil a výsledkem je právě *Novell2*. Více se o doprovodné disketě dozvíte v příloze B.

Za vaši pozornost stojí i webová adresa na serveru SPŠSE v Liberci `http://www.spsselib.hiedu.cz/~vais/sw.htm`, kde najdete některé mé (nejen) síťové programy, většinou včetně zdrojového textu v jazyce Turbo Pascal.

Závěr

Nyní jsme na samém konci publikace. Pokud jste ji svědomitě dočetli až sem, nemělo by vám činit problémy napsat nějaký program, který bude využívat rozhraní IPX. Jsem si vědom, že problematika není vůbec triviální. Než si IPX opravdu zažijete, bude to jistě nějaký čas trvat. Zkuste si nyní naprogramovat nějaké jednodušší aplikace, třeba hry s využitím rozhraní IPX. Možná budete překvapeni, jak to je jednoduché. Nemusíte totiž programovat inteligenci počítače, což bývá u podobných případu nejtěžší. Stačí naprogramovat uživatelské rozhraní, nadefinovat si formát komunikačních dat a je hotovo. „Člověče, nezlob se", „Piškvorky" nebo „Šibenici" tak naprogramujete za jedno odpoledne (pokud umíte programovat).

Přeji vám, abyste poznatky získané z této knihy v životě dobře uplatnili, a na jejich základě získali ještě spoustu dalších užitečných vědomostí na vaší případné budoucí dráze informatika.

Vladimír Váňa

Závěr DP

Cílem diplomové práce bylo vytvořit výukový text IPX pro střední školy. Tato problematika vypadá na první pohled velice obtížně a každého studenta asi ihned odradí, pokud si prohlédne nějakou publikaci, která o IPX pojednává. Jelikož ani u mne tomu nebylo jinak, snažil jsem se napsat učebnici tak, aby se student nezděsil už při letmém prolistování textu.

Mezi hlavní příčiny odpudivosti publikací o IPX patří dle mého mínění následující:

- Text je psán anglicky.
- Příklady jsou v jazyce C nebo dokonce v JSA.
- Příklady vypadají, jako by byly psány v JSA a nikoliv v Pascalu nebo C a tudíž jsou neúnosně dlouhé.
- Text je pouze referenční příručkou.
- V textu jsou osvětleny problémy, které jistě potěší odborníka, ale nikoliv začátečníka.

Já jsem se pokusil těmto problémům čelit a věřím, že se mi to alespoň z části podařilo.

Učebnici jsem napsal v českém jazyce. Pokud člověk neumí anglicky opravdu dobře a navíc o probírané problematice nic neví, je pro něho text v angličtině značně nesrozumitelný. Myslím si, že po nastudování problematiky z české literatury je práce s anglickou podstatně příjemnější, protože už čteme o něčem, co známe, na čemž se zároveň seznámíme s anglickými odbornými termíny, které se pak vyskytují i v částech, které studujeme, protože obsahují něco nového.

Příklady jsem vypracoval v jazyce Turbo Pascal. JSA je příliš obtížný a navíc velice neprůhledný – zatímco na pochopení středně složitého programu

v Pascalu stačí několik minut, na stejný program v JSA je potřeba několik hodin. Vzhledem k tomu, že v současné době se programování většinou vyučuje s pomocí jazyka Pascal, dal jsem mu přednost před jazykem C, který není ve svých konstrukcích tak názorný jako Pascal.

Abych minimalizoval délku příkladů a maximalizoval srozumitelnost, snažil jsem se co nejvíce „nezajímavého" programového kódu odsunout do jednotky. Při psaní vlastních programů se tak student může věnovat pouze nezbytně nutným úkonům, potřebným pro práci s IPX. Bez využití jednotky by se všechny příklady velmi znepřehlednily a navíc by se IPX stalo výsadou pouze jedinců se znalostmi systémového programování. Takhle může IPX využívat každý student, který se naučil programovat v Pascalu.

Při psaní demonstračních příkladů jsem preferoval srozumitelnost před efektivností. Raději provádím třikrát stejný výpočet v místě, kde je výsledek použit, než abych si na jiném místě hodnotu vypočítal jednou a pak ji použil prostřednictvím proměnné. U kratších opakujících se posloupností příkazů dávám přednost jejich přímému zápisu do programu v místech, kde jsou potřeba, než abych pro ně definoval zvláštní procedury. Ukázkové programy je tak možno studovat systematicky od začátku do konce a není třeba neustále ve zdrojovém textu hledat, co která proměnná obsahuje nebo co která procedura provádí.

Dalším neduhem komerčních publikací je, že se v nich pořádně nevysvětlí, jak se vlastně IPX používá, čímž je čtenář nucen, aby z chatrných příkladů zkoumal, „jak to vlastně funguje", příp. aby se zeptal někoho, kdo již má s IPX zkušenosti.

Jednotlivé procedury IPX bývají většinou řazeny zcela nevhodně v abecedním pořadí namísto vhodnějšího tematického řazení. Jejich popis je prováděn formou neotřesitelných faktů bez špetky lidskosti. Tato učebnice se nesnaží do studenta „nahustit" za každou cenu vše, co je o IPX známo a co IPX dovede.

Z tohoto důvodu jsou v textu vypuštěny některé funkce IPX, které mi nepřipadaly jako nezbytné a o jejichž využití v „normálních" programech mám silné pochybnosti. Tyto pasáže by dle mého mínění studentovi stejně příliš nedaly a akorát by v něm posilovaly pocit, že programování s IPX je velice obtížné. Je to jako kdyby se v učebnici fyziky na střední škole v části o elektřině probíraly Maxwellovy rovnice v diferenciálním tvaru nebo vliv kvantově mechanických veličin na elektrické vlastnosti látky.

K dalším nevýhodám publikací o IPX patří, že je psal odborník, který spoustu věcí považuje za samozřejmost a tudíž se nezdržuje jejich vysvětlováním. Já jsem se snažil napsat text pro úplné síťové nováčky. Z tohoto důvodu jsem také v první části zařadil pojednání o sítích a hlavně popis sítě Novell NetWare, místo abych např. detailně popisoval, jak si směrovače vyměňují směrovací informace a jak si z nich pak tvoří směrovací tabulky. Přijde mi, že autoři komerčních publikací se hlavě chtějí předvést před svými kolegy a vůbec si nepřipouští, že by jejich knihy mohl číst také někdo, kdo teprve s probíranou problematikou začíná. Možná na to mají trochu právo, oni přece nepíší učebnici. Ale já ano. Právě proto jsem postupoval při psaní přesně opačně, snažil jsem se vysvětlit hlavně základy. Kniha není na nijak vysoké odborné úrovni, ale o to víc je čtivější třeba i proto, že jsem se snažil uvádět, kde to jen šlo, (někdy i mírně humorné) příklady. Uznávám, že některé pasáže jsou nezáživné. Naštěstí jich není mnoho. Co mě trochu mrzí, je malé množství obrázků. Ty čtenáře vždycky potěší. Obsah této knihy bohužel nebyl vhodný k tematickému ilustrování a na efektní grafické prvky pro oživení textu, které se v současné době rozmáhají stále více, jsem si prostě netroufl. Jednak nevlastním potřebné technické a programové vybavení a jednak neumím kreslit. A kromě toho barevný tisk by náklady na vytištění knihy několikanásobně zvýšil.

Co říci na samém závěru? Bude to dobrá učebnice? Naučí se podle ní někdo něco? Byl bych moc rád, kdyby tomu tak bylo, ale na sto procent o tom

přesvědčen nejsem. Co se líbí jednomu, to jiného dráždí. Jemný humor v knize může někoho nutit k pláči, jiného k záchvatům smíchu. Domnívám se, že protokol IPX byl vysvětlen náležitě, příklady byly srozumitelné a názorné, čtenář se dozvěděl spoustu informací, které s IPX více či méně souvisí, ale stačí tohle všechno? Hlavní je zájem o danou problematiku a vhodné naservírování látky studentovi učitelem. S tím učebnice nic nepořídí. V dnešní době jsou ceny výpočetní techniky již poměrně nízké, proto si myslím, že ani nutnost provozovat kvůli výuce IPX počítačovou síť by neměla být překážkou k použití učebnice na střední škole. A věřím, že časem, až se učitelům upraví mzdové podmínky, budou i na středních školách na informatiku kvalifikovaní učitelé z pedagogických fakult a nikoliv důchodci, kteří někdy v dřívějším zaměstnání (možná) pracovali s počítačem. Z vlastní zkušenosti vím, že hlavně v menších městech jsou na tom střední školy s humanitním zaměřením, co se týče informatiky, velice bídně, a právě proto bych chtěl jít na jedno takové menší humanitní gymnázium učit.

Přílohy

A Jednotka VVIPX

```
unit VVIPX;
interface
uses dos;
type
  NodeId = array [1..6] of byte;
  NetwId = array [1..4] of byte;
  IPXAddress = record
                 Network: NetwId;
                 Node: Nodeid;
                 Socket: Word;
                 end;
  IPXhead = record
              CheckSum: array[1..2] of byte;
              Length: word;
              TransCtrl: byte;
              PacketTyp: byte;
              DestNetw: Netwid;
              DestNode: NodeId;
              DestSckt: word;
              SrceNetw: Netwid;
              SrceNode: NodeId;
              SrceSckt: word;
            end;
  ECBblock = record
```

```
        Link: array[1..2] of integer;
        EsrAddr: pointer;
        InUse: byte;
        ComplCode: byte;
        ScktNum: array[1..2] of byte;
        IPX_w_spc: array[1..4] of byte;
        Driver_sp: array[1..12] of byte;
        ImmdAddr: NodeId;
        FragCnt: array[1..2] of byte;
        FragAdr: pointer;
        FragSize: word;
      end;
const vsem: nodeid =($FF, $FF, $FF, $FF, $FF, $FF);
      unknown: netwid =(0, 0, 0, 0);
      ShortLive = 0;
      LongLive = $FF;
      OpenStyle: byte = ShortLive;
      Successful = $00;
      IPX_NotInstalled = $F0;
      RequestCancelled = $FC;
      BadPacket = $FD;
      PacketNotDeliverable = $FE;
      SocketTableFull = $FE;
      SocketAlreadyOpen = $FF;
      SocketClosed = $FF;
      HardwareFailure = $FF;
Procedure IPXInitialize;
Procedure OpenSocket(number: word);
```

```
Procedure CloseSocket(number: word);
Procedure GetLocalTarget(networkAddress: IPXAddress; var immAddress: no
          var TransportTime: Word);
Procedure SendPacket(var ECB: ECBblock);
Procedure ReceivePacket(var ECB: ECBblock);
Procedure CancelEvent(var ECB: ECBblock);
Procedure GetMyNetAddr(var NodeNum: nodeid; var Netwnum: netwid);
Procedure GetInternetAddress(connectionNumber: Byte;
          var networkNumber: netwid; var physicalNodeAddress: nodeid);
var NetError: byte;
implementation
var nebeziSit: Boolean;
    r: Registers;
    Int7A: pointer;
    IPXCall: pointer;
Procedure CallNetw(rr: Registers);
var aa: word;
    _AX, _BX, _CX, _DX, _BP, _SI, _DI, _DS, _ES, Flags: Word;
begin
_AX := r.AX;
_BX := r.BX;
_CX := r.CX;
_DX := r.DX;
_BP := r.BP;
_SI := r.SI;
_DI := r.DI;
_DS := r.DS;
_ES := r.ES;
```

```
Flags := r.Flags;
asm
        jmp @@hop
@@gg: dd 0
@@bp: dw 0
@@hop:push ax
        push bx
        push cx
        push dx
        push si
        push di
        push es
        push ds
        mov word ptr @@BP, BP
        mov ax, word ptr [IPXCall]
        mov bx, word ptr [IPXCall+2]
        mov word ptr [@@gg], ax
        mov word ptr [@@gg+2], bx
        mov AX, _AX
        mov BX, _BX
        mov CX, _CX
        mov DX, _DX
        mov SI, _SI
        mov DI, _DI
        mov ES, _ES
        mov DS, _DS
        mov BP, _BP
        call dword ptr cs:[@@gg]
```

```
        push bp
        mov BP, word ptr @@BP
        mov _AX, AX
        mov _BX, BX
        mov _CX, CX
        mov _DX, DX
        pop AX
        mov _BP, AX
        mov _SI, SI
        mov _DI, DI
        mov _ES, ES
        mov _DS, DS
        pop ds
        pop es
        pop di
        pop si
        pop dx
        pop cx
        pop bx
        pop ax
    end;
  r.AX := _AX;
  r.BX := _BX;
  r.CX := _CX;
  r.DX := _DX;
  r.BP := _BP;
  r.SI := _SI;
  r.DI := _DI;
```

```
r.DS := _DS;
r.ES := _ES;
r.Flags := Flags;
end;
Procedure OpenSocket(number: word);
begin
if nebeziSit then begin NetError := $F0; exit; end;
r.bx := 0;
r.al := OpenStyle;
r.Dx := number;
CallNetw(r);
NetError := R.AL;
end;
Procedure IPXOpenSocket(var number: word);
begin
if nebeziSit then begin NetError := $F0; exit; end;
r.bx := 0;
r.al := OpenStyle;
r.Dx := number;
CallNetw(r);
if number = 0 then number := r.Dx;
NetError := R.AL;
end;
Procedure CloseSocket(number: word);
begin
if nebeziSit then begin NetError := $F0; exit; end;
r.BX := 1;
r.DX := number;
```

```
CallNetw(r);
NetError := R.AL;
end;
Procedure GetLocalTarget(networkAddress: IPXAddress; var immAddress: no
            var TransportTime: Word);
type GetLTtyp = record
      addr: IPXAddress;
      imadr: nodeid;
    end;
var Glt: GetLTtyp;
begin
if nebeziSit then begin NetError := $F0; exit; end;
Glt.addr := networkAddress;
r.BX := 2;
r.ES := Seg(Glt);
r.SI := ofs(Glt.imadr);
r.DI := ofs(Glt.addr);
Intr($7a, R);
NetError := R.AL;
TransportTime := R.CX;
immAddress := Glt.imadr;
end;
Procedure SendPacket(var ECB: ECBblock);
var delka: word;
begin
if nebeziSit then begin NetError := $F0; exit; end;
r.BX := 3;
r.ES := Seg(ECB);
```

```
r.SI := ofs(ECB);
CallNetw(r);
NetError := R.AL;
end;
Procedure ReceivePacket(var ECB: ECBblock);
var delka: word;
begin
if nebeziSit then begin NetError := $F0; exit; end;
r.BX := 4;
r.ES := Seg(ECB);
r.SI := ofs(ECB);
CallNetw(r);
NetError := R.AL;
end;
Procedure CancelEvent(var ECB: ECBblock);
begin
if nebeziSit then begin NetError := $F0; exit; end;
r.BX := 6;
r.ES := Seg(ECB);
r.SI := ofs(ECB);
CallNetw(r);
NetError := R.AL;
end;
procedure GetMyNetAddr(var NodeNum: nodeid;var Netwnum: netwid);
type InternetworkAddress = record
                             Netw: NetwId;
                             Node: Nodeid;
                           end;
```

```
var Reply:InternetworkAddress;
begin
if nebeziSit then begin NetError := $F0; exit; end;
R.ES := Seg(Reply);
R.SI := Ofs(Reply);
R.BX := 9;
CallNetw(r);
NodeNum := Reply.Node;
NetwNum := Reply.Netw;
NetError := R.AL;
end;
Procedure GetInternetAddress(connectionNumber: Byte;
            var networkNumber: netwid;
            var physicalNodeAddress: nodeid);
type SendForGetAddr = record
            request: Word;
            r2: Byte;
ConNum: Byte;
          end;
     RecvForGetAddr = record
            request: Word;
            network: Netwid;
            node: Nodeid;
            socket: Word;
          end;
var
 ccode: Word;
 sendPacket: SendForGetAddr;
```

```
 receivePacket: RecvForGetAddr;
begin
sendPacket.r2 := 19;
sendPacket.ConNum := connectionNumber;
sendPacket.Request := 2;
receivePacket.Request := 12;
R.AH := 227;
R.DS := Seg(sendPacket);
R.SI := Ofs(sendPacket);
R.ES := Seg(receivePacket);
R.DI := Ofs(receivePacket);
intr($21, r);
NetError := R.AL;
networkNumber := receivePacket.network;
physicalNodeAddress := receivePacket.node;
end;
Procedure IPXInitialize;
var je: byte;
begin
 asm
        push ES
        mov DI, 0
        mov AX, 7A00h
        int 2Fh
        mov je, AL
        mov word ptr [IPXCall], DI
        mov word ptr [IPXCall+2], ES
        pop ES
```

```
 end;
 nebeziSit := je <> 255;
 if nebeziSit then NetError := $F0
              else NetError := 0;
end;
begin
  IPXInitialize;
end.
```

B Disketa

Na přiložené disketě najdete všechny ukázkové programy, jednotku VVIPX a jednotky z Internetu. Dále pak soubory s informacemi, které se vztahují k probírané problematice a vybrané programy ze stránky `http://www.spsselib.hiedu.cz/~vais/sw.htm`, o které se zmiňuji na straně 90.

Literatura

[1] Jaroslav Fojtík, Václav Valtr: *NetBIOS a IPX/SPX, programování s využitím síťového rozhraní.*
Grada, Praha 1993

[2] Barry Nance: *Network Programming in C.*
Que Corporation, Carmel 1990

[3] Oldřich Přichystal: *Téměř vše o sítích Novell.*
Grada, Praha 1992

[4] Pavel Satrapa: *Počítačové sítě, přednášky k předmětu.*
Vladimír Vais, Liberec 1996

[5] Pavel Satrapa, Jiří A. Randus: *LINUX – Internet server.*
Neokortex, Praha 1996

[6] Petr Olšák: *Typografický systém TEX.*
Československé sdružení uživatelů TEXu, Praha 1995

[7] Jan Havelka: *Počítačová typografie pro každého.*
Grada, Praha 1995

[8] Martin Kvoch: *Programování v Turbo Pascalu 7.0.*
Kopp, České Budějovice 1995

Rejstřík