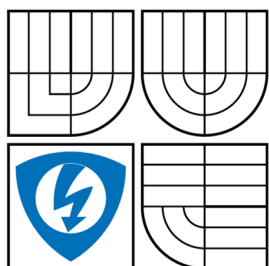


VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH
TECHNOLOGIÍ

ÚSTAV TELEKOMUNIKACÍ

FACULTY OF ELECTRICAL ENGINEERING AND COMMUNICATION
DEPARTMENT OF TELECOMMUNICATIONS

TVORBA SÍŤOVÉHO OVLADAČE PRO RODINU OPERAČNÍCH SYSTÉMŮ WINDOWS

DIPLOMOVÁ PRÁCE

DIPLOMA THESIS

AUTOR PRÁCE

AUTHOR

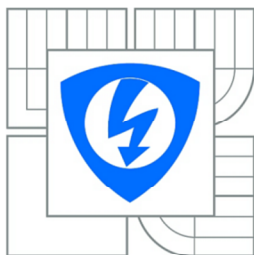
Bc. Radko KRKOŠ

VEDOUCÍ PRÁCE

SUPERVISOR

Ing. Lukáš RŮČKA

BRNO 2011



VYSOKÉ UČENÍ
TECHNICKÉ V BRNĚ

Fakulta elektrotechniky
a komunikačních technologií

Ústav telekomunikací

Diplomová práce

magisterský navazující studijní obor
Telekomunikační a informační technika

Student: Bc. Radko Krkoš

ID: 78626

Ročník: 2

Akademický rok: 2010/2011

NÁZEV TÉMATU:

Tvorba síťového ovladače pro rodinu operačních systémů Windows

POKYNY PRO VYPRACOVÁNÍ:

Podrobně se seznámte s funkcí ovladačů v operačních systémech rodiny Windows (Windows XP, Windows Vista, Windows 7) a s principy tvorby těchto ovladačů. Zaměřte se především na ovladače síťového rozhraní. Popište možné postupy při tvorbě ovladačů síťového rozhraní s ohledem na možnou následnou funkční implementaci ovladače. Implementujte ovladač pro verzi operačního systému Windows s verzí jádra 6.x. Tato implementace by měla umožňovat manipulaci se záhlavím IP paketu podle uživatelského nastavení. Manipulace musí zahrnovat především změny v poli DSCP v záhlaví IP paketů, dle uživatelem definovaných parametrů (jako je např. protokol, IP adresa, port apod.). Výsledná implementace musí mít jasně definované rozhraní a možnost uživatelsky nastavitelného chování. Celý postup implementace a výsledný ovladač podrobně zdokumentujte.

DOPORUČENÁ LITERATURA:

[1] Russinovich, M., Solomon, D.: Windows Internals: Including Windows Server 2008 and Windows Vista. Washington: Microsoft Press, 5. edice, 2009. 1264 s. ISBN: 978-0735625303.

[2] Kadlec, V.: Učíme se programovat v jazyce C. Praha: Computer Press, 2002. 277 s. ISBN 80-7226-715-9.

[3] Harbison, S., Steele, G.: Referenční příručka jazyka C. Veletiny: Science, 1996. 334 s. ISBN: 80-901475-50.

Termín zadání: 7.2.2011

Termín odevzdání: 26.5.2011

Vedoucí práce: Ing. Lukáš Růčka

prof. Ing. Kamil Vrba, CSc.

Předseda oborové rady

Abstrakt

Práca popisuje postup návrhu a tvorby ovládača pre značkovanie sieťovej premávky kvôli zabezpečeniu kvality služby. Na začiatku sú popísané a porovnané rôzne technológie použiteľné pre tvorbu takéhoto ovládača a následne je vybraný vhodný kandidát pre použitie. Ďalej sa práca venuje návrhu ovládača podľa zadania a vytvára podrobnejší súbor požiadaviek. Tie ovplyvňujú samotný dizajn ovládača. Pri návrhu sa práca sústreďuje na kritériá ako minimalizácia vplyvu ovládača na parametre sieťovej premávky a minimálna záťaž na systém. Okrem ovládača je vytvorené aplikačné rozhranie pre jeho riadenie z režimu používateľa a jedná sa tak o systém úplne zabezpečujúci značkovanie premávky. V závere je popísaná vytvorená implementácia ovládača a aplikačné rozhranie z pohľadu programátora aplikácií využívajúcich poskytnuté rozhranie.

Kľúčové slová

Windows, ovládač, WFP, QoS značkovanie

Abstract

This thesis describes the approach in designing and implementing a driver marking network traffic for quality of service provision. In the beginning distinct technologies usable for the driver development are discussed and compared to each other while the most suitable candidate is chosen for driver development. Next the thesis discusses the design of the driver according to specification and depicts a list of more specific requirements affecting the driver design itself. The focus point in design is the least possible impact on network traffic performance and the running system itself. In addition to the driver an application programming interface usable from user mode is designed and so the system as a whole ensures the marking of network traffic entirely. In the end the created driver architecture is described and the application programming interface is described in terms of use.

Keywords

Windows, driver, WFP, QoS marking

Bibliografická citácia

KRKOŠ, R. *Tvorba síťového ovladače pro rodinu operačních systémů Windows*. Brno: Vysoké učení technické v Brně, Fakulta elektrotechniky a komunikačních technologií, 2011. 81 s. Vedúci bakalárskej práce Ing. Lukáš Růčka.

PREHLÁSENIE

Prehlasujem, že svoju diplomovú prácu na tému „Tvorba síťového ovladače pro rodinu operačních systémů Windows“ som vypracoval samostatne pod vedením vedúceho diplomovej práce a s použitím odbornej literatúry a ďalších informačných zdrojov, ktoré sú všetky citované v práci a uvedené v zozname použitej literatúry na konci práce.

Ako autor uvedeného semestrálneho projektu ďalej prehlasujem, že v súvislosti s vytvorením tejto práce som neporušil autorské práva tretích osôb, najmä som nezasiahol nedovoleným spôsobom do cudzích autorských práv osobnostných a som si úplne vedomý následkov porušenia ustanovenia § 11 a nasledujúcich autorského zákona č. 121/2000 Zb. platného v Českej republike, vrátane možných trestnoprávných dôsledkov vyplývajúcich z ustanovenia § 152 trestného zákona č. 140/1961 Zb. platného v Českej republike.

V Brne

.....

(podpis autora)

Obsah

Zoznam obrázkov	8
Zoznam tabuliek	9
Úvod	10
1 Literárna rešerš diplomovej práce	11
1.1 Súčasný stav problematiky	11
1.1.1 Tvorba ovládača OS Windows pre filtrovanie sieťovej premávky	11
1.1.2 Tvorba API pre režim používateľa	12
1.1.3 Kvalita služby	13
2 Teoretický rozbor problému	14
2.1 Ovládače zariadení	14
2.2 Typy a modely ovládačov OS Windows	15
2.2.1 Windows driver model	17
2.2.2 Kernel-Mode Driver Framework	17
2.2.3 User-Mode Driver Framework	17
2.2.4 Modely špecifické pre konkrétny typ zariadenia	18
2.3 Vrstvová architektúra WDM	18
2.3.1 Ovládače zberníc	19
2.3.2 Funkčné ovládače	19
2.3.3 Filtre	19
2.3.4 Triedy zariadení OS Windows	20
2.4 Štruktúra ovládača OS Windows	21
2.5 Sieťové ovládače	22
2.5.1 Rozdiely medzi systémami NT 5.x a NT 6.x	23
2.6 Filtrovanie sieťovej premávky	24
2.6.1 Transport driver interface	26
2.6.2 Network driver interface specification	26
2.7 Windows filtering platform	29
2.7.1 Architektúra WFP	29
2.8 Technológie tretích strán	31

2.8.1	CACE Technologies WinPcap	31
2.8.2	NT Kernel WinpkFilter 3.0	32
2.9	Praktické poznámky k tvorbe ovládačov	32
2.9.1	Zostavenie ovládača	33
2.9.2	Inštalácia ovládača	34
2.9.3	Súbory INF pre ovládače	35
2.9.4	Podpisovanie ovládača	36
2.10	Zhodnotenie technológií	37
3	Zvolená metodika riešenia	38
3.1	Štruktúra ovládača WFP	38
3.2	Pravidlá pre filtrovanie sieťovej premávky.....	39
3.3	Návrh štruktúry ovládača pre filtrovanie premávky	41
3.4	Optimalizácia náročnosti ovládača.....	41
3.4.1	Vplyv ovládača na oneskorenie paketu a kolísanie oneskorenia.....	42
3.4.2	Vplyv ovládača na stratovosť paketov a zmenu ich poradia.....	42
3.4.3	Ovplyvniteľné sekcie ovládača	43
3.4.4	Vyhľadávací algoritmus	44
3.4.5	Spracovateľské vlákno.....	44
3.4.6	Pamäťová zložitosť	45
3.4.7	Meranie vplyvu ovládača na parametre sieťovej premávky.....	46
3.5	Aplikačné rozhranie pre riadenie ovládača	46
3.5.1	Technické prostriedky aplikačného rozhrania	47
3.5.2	Štruktúra hlavičkového súboru API.....	48
4	Sieťový ovládač pre značkovanie premávky	49
4.1	Použitie vytvoreného ovládača	49
4.2	Ovládač pre značkovanie paketov	49
4.2.1	Vstupná a výstupná funkcia ovládača	49
4.2.2	Podporná logika ovládača	52
4.2.3	Filtračná logika ovládača	56
4.3	Aplikačné rozhranie ovládača.....	59
4.3.1	Funkcie aplikačného rozhrania pre riadenie ovládača.....	60

4.3.2	Funkcie pre načítanie a uloženie pravidiel.....	61
4.3.3	Funkcie pre prácu s pravidlami	62
4.3.4	Funkcie pre riadenie ovládača	63
4.3.5	Funkcie pre získanie hodnôt parametrov pravidiel	63
4.3.6	Funkcie pre spracovanie používateľských dát.....	64
4.3.7	Funkcie pre získanie štandardných hodnôt parametrov	65
4.3.8	QoS Management Console.....	66
5	Záver.....	68
	Použitá literatúra.....	69
	Zoznam použitých skratiek.....	71
	Prílohy.....	76

Zoznam obrázkov

Obr. 2.1 - Zaradenie ovládačov zariadení v architektúre OS.....	14
Obr. 2.2 - Rozdelenie typov ovládačov OS Windows	16
Obr. 2.3 - Vrstvová architektúra WDM	18
Obr. 2.4 - Architektúra NDIS.....	27
Obr. 2.5 - Architektúra Windows Filtering Platform	30
Obr. 2.6 - Voľné zostavenie ovládača pre Windows 7 x86	34
Obr. 3.1 - Zjednodušená štruktúra ovládača pre filtrovanie sieťovej premávky	41
Obr. 3.2 - Vrstvy aplikačného rozhrania pre riadenie ovládača.....	47
Obr. 3.3 - Stavový automat načítania REG súboru.....	48
Obr. 4.1 - Štruktúra funkcie pre inicializáciu ovládača.....	51
Obr. 4.2 - Štruktúra funkcie pre odstránenie ovládača.....	52
Obr. 4.3 - Štruktúra funkcie pre načítanie pravidiel.....	53
Obr. 4.4 - Štruktúra funkcie pre registráciu popisov.....	53
Obr. 4.5 - Štruktúra funkcie pre registráciu popisných ovládačov pre transportnú vrstvu.....	55
Obr. 4.6 - Funkcia pre registráciu filtrov	55
Obr. 4.7 - Štruktúra funkcie pre zrušenie registrácie popisu	56
Obr. 4.8 - Štruktúra funkcie pre klasifikáciu.....	57
Obr. 4.9 - Štruktúra funkcie pre dokončenie vloženia paketu	58
Obr. 4.10 - Štruktúra funkcie pre značkovanie paketov	58
Obr. 4.11 - Štruktúra spracovateľského vlákna.....	59
Obr. 4.12 - Hlavné okno QoS Management Console	66
Obr. 4.13 - Formulár pre pridanie nového pravidla	67

Zoznam tabuliek

Tab. 2.1 - Zmeny v metódach spracovania sieťovej premávky medzi verziou jadra 5.x a 6.x .	23
Tab. 2.2 - Verzie NDIS podporované v operačných systémoch	27
Tab. 2.3 - Možnosti licencií NT Kernel WinpkFilter	32
Tab. 2.4 - Komponenty balíčka ovládača.....	35
Tab. 2.5 - Možnosti INF súborov pre inštaláciu zariadení.....	36
Tab. 3.1 - Zoznam procedúr ovládača a ich využitie WFP ovládačom	38
Tab. 3.2 - Funkcie spätného volania ovládačov Windows Filtering Platform	39
Tab. 3.3 - Parametre pravidla pre značkovanie sieťovej premávky	40

Úvod

Cieľom práce je vytvorenie sieťového ovládača filtrujúceho sieťovú premávku a značujúceho pakety značkami pre zabezpečenie kvality služby podľa používateľsky definovaných pravidiel.

Prvá časť práce je literárnou rešeršou. Popisujú sa tu použité zdroje a ich využitie pri vypracovávaní práce. Takisto je to zhodnotenie použitých materiálov pre štúdium tematiky. Použité zdroje sú rozdelené do niekoľkých častí podľa tematických okruhov a zamerania na jednotlivé časti praktickej práce. Prvá časť poslúži pri rozširovaní teoretických vedomostí o problematike.

Druhá časť práce obsahuje teoretickú analýzu riešeného problému. Popisuje funkciu ovládačov, špeciálne sieťových ovládačov, v operačných systémoch rodiny Microsoft Windows. Cieľom je porovnanie technológií použiteľných pre tvorbu ovládača so zameraním v konečnej fáze implementovať funkčný ovládač pre značkovanie sieťovej premávky pre zabezpečenie kvality služby. V tejto časti sú popísané rôzne zvažované technológie a porovnávané ich výhody a nevýhody, časť vo veľkej miere vychádza z teoretickej analýzy vykonanej v práci predchádzajúcom semestrálnom projekte. V závere druhej časti sú učené odporúčania technológie pre využitie pri tvorbe ovládača. Druhá časť zhrňa rozličné technológie pre tvorbu sieťových ovládačov.

V tretej časti je popisovaný proces návrhu výsledného systému a sú tu diskutované rozličné požiadavky, ktoré by mal spĺňať. Je tu navrhnutá štruktúra pravidiel pre filtrovanie sieťovej premávky a tiež špecifikácia rozhrania pre riadenie ovládača. Rozhranie je vytvorené tak, aby spĺňalo určené požiadavky, vyplývajúce zo zadania a plánovaného použitia systému. Tretia časť popisuje dôvody vedúce k štruktúre systému, ktorá bola navrhnutá.

V záverečnej štvrtej časti je popísaná samotná implementácia návrhu. Systém ako celok je rozdelený na tri oddelené, ale spolu súvisiace časti. Je tu popísaná vnútorná štruktúra ovládača, aplikačné rozhranie pre riadenie ovládača najmä zo stránky použitia a aplikácia využívajúca toto rozhranie, umožňujúca správu pravidiel. Štvrtá časť popisuje konečný výsledok práce a možnosti jeho využitia.

1 Literárna rešerš diplomovej práce

Kvôli tomu, že práca je rozdelená do viacerých častí, ktoré spolu po technickej stránke nesúvisia ale tiež vzhľadom k tomu, že niektoré časti vyžadujú hlbšie teoretické znalosti, nie je problematika práce popísaná centralizovane v konkrétnom zdroji. Preto pri tvorbe práce bolo čerpané z viacerých zdrojov.

1.1 Súčasný stav problematiky

Použitú literatúru je možné z hľadiska tematiky práce rozdeliť do niekoľkých častí:

- popis architektúry OS (operačný systém) Windows všeobecne a so zameraním na tvorbu ovládačov;
- realizácia algoritmov v režime používateľa a podporné funkcie OS pre programy v režime používateľa;
- programovanie aplikácií v režime používateľa.

Toto je hrubé delenie, niektorá literatúra spadá do viacerých kategórii, podobne popis kategórií nie je vyčerpávajúcou špecifikáciou.

1.1.1 Tvorba ovládača OS Windows pre filtrovanie sieťovej premávky

Z pohľadu tvorby ovládačov OS Windows bol najcennejším zdrojom [1], ktorý do hĺbky popisuje architektúru OS Windows verzie jadra 6.x. Umožňuje získať teoretické znalosti, ktoré sú nutne potrebné pre realizáciu systémov pracujúcich v režime jadra, ktorými ovládače sú. Staršie vydanie tejto publikácie [2] bolo využité najmä v začiatkoch práce, pred získaním novej verzie, ale tiež umožňuje lepšie pochopenie rozsiahlych zmien a zlepšení v jadre OS Windows medzi verziami 5.x a 6.x, z pohľadu práce najmä výrazných zmien v sieťovom zásobníku.

Dobрым zdrojom technických informácií o architektúre OS Windows je [3]. Taktiež popisuje jednotlivé volania OS a preto funguje ako príručka pri praktickej práci.

Ďalšie tri použité zdroje slúžia na získanie hlbších teoretických informácií o konkrétnych subsystémoch OS Windows. Zdroj [4] popisuje vrstvu abstrakcie zariadení a jej význam v OS, [5] sa venuje Windows Filtering Platform, vo verzii jadra 6.x novej technológii pre filtrovanie sieťovej premávky, ktorá je priamo súčasťou sieťového zásobníka OS. Popisuje jej architektúru a základné možnosti využitia, ale tiež poskytuje základné informácie o tom, ktoré staršie technológie nahrádza a aké výhody oproti nim ponúka. Technické informácie o WFP (platforma pre filtrovanie vo Windows) a ponúkaných volaniach sú obsiahnuté v [6], ktorý je veľmi vhodnou príručkou.

Článok [7] popisuje podrobnosti ohľadom stránkovanej a nestránkovanej pamäte, ich význam, rozdieloch a systémových obmedzeniach najmä nestránkovanej pamäte.

Winsock SPI (rozhranie pre poskytovateľov služieb), jedna z technológií využívaných pred novým sieťovým zásobníkom¹ pre tvorbu nastavieb transportnej vrstvy, poskytujúcich pridané služby, je popísaná v [8], pričom [9] je ďalej špecializovaný na časti priamo využiteľné pre filtrovanie sieťovej premávky.

Architektonické rozhodnutia pri tvorbe aplikácií typu firewall pre rôzne verzie OS Windows sú popísané v [10]. Systémy typu firewall sú príbuzné téme práce a ich analýzy sú preto vhodným zdrojom informácií.

Keďže WFP je v čase tvorby práce relatívne nová technológia, tak v nedávnej minulosti trpela vážnymi chybami. Tie z nich viažuce sa k práci sú popísané v [11], [12] a [13]. Ďalšie chyby, pre ktoré ešte neboli vydané opravy, ale aj spresnenia formálnej dokumentácie a odbornú diskusiu k WFP obsahuje [14].

Ďalšie dva zdroje popisujú iné zvažované, ale nakoniec nevyužité technológie. V [15] je popis knižnice WinPCap, implementácie knižnice libpcap z OS UNIX (viac úlohový a viac používateľský operačný systém) a kompatibilných. Zdroj [16] popisuje knižnicu WinkFilter, platené API na tvorbu aplikácií pre filtrovanie sieťovej premávky od NT (nová technológia) Kernel Resources.

1.1.2 Tvorba API pre režim používateľa

Kvôli architektúre vytváraného systému je API (rozhranie pre programovanie aplikácií) programom čisto pre režim používateľa. Pre jeho tvorbu bol použitý jazyk C++ (objektovo orientovaný programovací jazyk). Vhodným zdrojom informácií o C++ je [17]. Táto publikácia má svoje počiatky v dobe vzniku štandardu C++ a popisuje ho dostatočne do hĺbky. Zameraná je na jazyk samotný, štandardná knižnica je popisovaná v menšom rozsahu. To kompenzuje [18], ktorá je vyčerpávajúcim popisom štandardnej knižnice jazyka C++. Obsahuje popis jednotlivých dátových kontajnerov a vstavaných algoritmov, ale venuje sa aj internacionalizácii a spracovaniu textových reťazcov.

Pre grafické používateľské rozhranie bolo využité MFC (základňové triedy spoločnosti Microsoft) – súbor tried od spoločnosti Microsoft pre zjednodušenie tvorby aplikácií pre OS Windows. Popis MFC je možné nájsť napr. v [19]. Kniha je štruktúrovaná na časti venujúce sa rôznym typom problémov a ponúka riešenia alebo návody ku riešeniam prehľadnou cestou. Ďalším vhodným zdrojom je slávna [20], venujúca sa širokej problematike do značnej hĺbky. Okrem tvorby aplikácií je tu možné nájsť aj informácie o tvorbe DLL (dynamicky linkované knižnice) knižníc. Pri programovaní s využitím MFC sa jedná o veľmi cennú príručku.

Zobierané poznatky a odporúčenia k tvorbe API k ovládačom je možné nájsť v [21]. Autor uvádza niekoľko pravidiel, ktoré zlepšia návrh a využitie API a navrhuje formu API, ktorá bude prenositeľná na čo najväčší počet systémov.² Pokračovanie [22] sa zameriava na

¹ termín nový sieťový zásobník (angl. New Network Stack) sa používa ako označenie sieťového zásobníka v OS Windows verzie jadra 6.x kvôli celkovému prepracovaniu tohto subsystému vedúcemu k značnému zlepšeniu v oblasti výkonu a použiteľnosti

² vývojových systémov

správny návrh rozhrania API a spôsoby, ako ho exportovať vývojovým prostrediam. V tretej časti [23] sa zameriava na praktické časti exportu rozhrania API používateľom. Ako celok tento seriál zahrňuje informácie potrebné k návrhu jednoduchého a udržateľného rozhrania v DLL knižnici exportujúcej čisté funkcie jazyka C (procedurálny programovací jazyk). Podrobnejšie informácie o DLL knižniciach, ich význame a použití sú uvedené v [24]. Technické informácie o dynamickom pripájaní knižníc počas behu programu sú uvedené v [25].

Popis štruktúry REG (súbor obsahujúci záznamy pre systémové registre) súborov a možnosti ich využitia sú dokumentované v [26] spolu s popisom distribúcie registrových záznamov na cieľové systémy.

Zdroj [27] obsahuje popis nástroja IPERF (nástroj pre meranie priepustnosti dátovej siete), slúžiaceho na testovanie priepustnosti spojenia medzi dvoma bodmi v dátovej sieti. Distribuovaný generátor umelej sieťovej premávky D-ITG (distribuovaný medzi sieťový generátor premávky) je zase popísaný v [28].

1.1.3 Kvalita služby

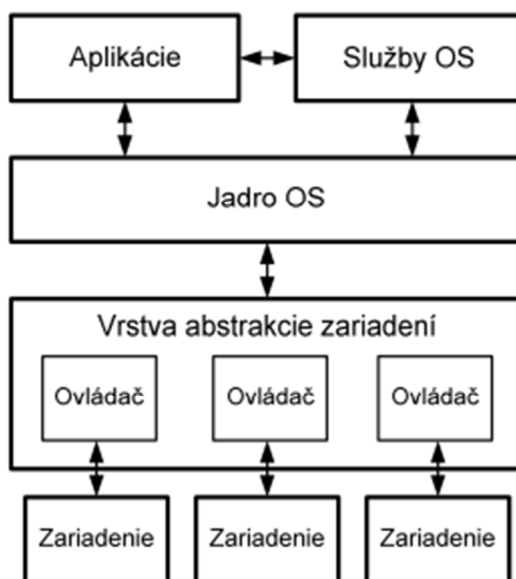
Aj napriek blízkej súvislosti práce so zabezpečením kvality služieb sú takéto teoretické poznatky využívané len málo. Preto v práci stačilo čerpať len z jedného zdroja, ktorým bol [29]. Tento zdroj podrobne popisuje zabezpečenie kvality služieb v počítačových sieťach všeobecne, Integrované služby, Diferencované služby, ale obsahuje aj hlbšie informácie, napr. o postupe klasifikácie a značkovania paketov.

2 Teoretický rozbor problému

V kapitole je popísaná funkcia ovládačov a ich zaradenie v architektúre operačného systému Microsoft Windows, s prihliadnutím na sieťové ovládače. Popis sa zameriava na štruktúru, ktorej musia ovládače pre OS Windows zodpovedať. Ďalej je popis smerovaný na filtrovanie sieťovej premávky a rôzne technológie použiteľné pre filtrovanie. Niektoré z týchto technológií nie je možné na Windows s verziou jadra 6.x použiť vôbec a boli nahradené inou technológiou. Iné použiť možné kvôli spätnej kompatibilitate je, ale nie je to odporúčané, pretože využitie novšieho postupu umožní tvorbu výkonnejšieho riešenia, ktoré môže využívať nové vlastnosti sieťového zásobníka, za kratší čas. Krátko sú popísané ďalšie zvažované technológie spolu so zhodnotením dôvodov, prečo neboli pre riešenie vybrané.

2.1 Ovládače zariadení

Ovládač zariadenia funguje ako prispôsobovacia medzivrstva medzi zariadením a OS, ktorým je toto zariadenie využívané. Každé zariadenie potrebuje vlastný ovládač prekladajúci generické úlohy na povely špecifické pre konkrétne zariadenie. Zaradenie ovládačov zariadení v architektúre OS je zobrazené na obr. 2.1.



Obr. 2.1 - Zaradenie ovládačov zariadení v architektúre OS

Ovládač zariadenia - načítateľný modul režimu jadra prepájajúci I/O systém a zodpovedajúci HW (hardware), umožňujúci programom vyššej úrovne pracovať s HW. V OS Windows ovládače zariadení nenarábajú so zariadeniami priamo, ale namiesto toho volajú časti HAL (vrstva abstrakcie zariadení), ktorá pracuje s HW [2]. Ovládač zjednodušuje programovanie aplikácií, pretože umožňuje písať kód nezávisle na konkrétnom zariadení, s ktorým bude aplikácia pracovať.

Vrstva abstrakcie zariadení (HAL) - programová vrstva pracujúca priamo s fyzickými súčasťami počítača. Pretože HAL pracuje na úrovni medzi HW a výkonnými zložkami OS, aplikácie a ovládače zariadení si nemusia byť vedomé špecifik konkrétnych fyzických súčastí. Vrstva abstrakcie zariadení umožňuje jednému ovládaču zariadenia pracovať so zariadením na rôznych kombináciách fyzických komponentov, čo vývoj ovládačov zariadení výrazne zjednodušuje. HAL skrýva špecifické detaily fyzických komponentov³ a keďže aplikáciám a ovládačom zariadení nie je povolený priamy prístup k fyzickým súčastiam, tak HAL umožňuje rovnaký prístup k rôznym fyzickým konfiguráciám [4].

Rozhranie zariadenia - funkčnosť zariadenia, ktorú ovládač vystavuje aplikáciám a ostatným súčastiam systému, je členom triedy rozhrania zariadenia definovanej v systéme, alebo určenej dodávateľom zariadenia. Ovládač môže poskytovať inštancie žiadnej, jednej alebo viacerých tried rozhrania zariadenia [3].

Aby bolo možné ovládač zariadenia integrovať s manažérom a ostatnými komponentmi I/O (vstupno-výstupný) subsystému, musí ovládač zodpovedať implementačným pravidlám špecifickým pre typ zariadenia, ktoré ovládač spravuje a pre rolu, akú hrá v správe tohto zariadenia [1].

2.2 Typy a modely ovládačov OS Windows

Windows podporujú širokú škálu typov ovládačov a programovacích prostredí. Dokonca pre jeden typ ovládača je možné použiť rôzne programovacie prostredia v závislosti na špecifickom type zariadenia, pre ktoré je ovládač určený. Najširšou kategorizáciou je rozdelenie ovládačov na ovládače v režime používateľa a ovládače v režime jadra [1].

Ovládače v režime používateľa:

- **virtuálne ovládače zariadení (angl. „virtual device drivers“)** - sú používané pri emulácii 16-bitových aplikácií OS MS-DOS⁴ (diskový operačný systém spoločnosti Microsoft.) Obsadia referencie na I/O porty používané programami MS-DOS a prekladajú ich na natívne funkcie I/O subsystému OS Windows, ktoré následne používajú skutočné ovládače konkrétnych zariadení. Keďže Windows sú plne chráneným OS a aplikácie MS-DOS bežia v režime používateľa, nemôžu priamo pristupovať k HW, ale musia na to používať ovládače bežiacie v režime jadra.
- **ovládače tlačiarň (angl. „printer drivers“)** - prekladajú grafické požiadavky nezávislé na zariadení na príkazy pre konkrétnu tlačiareň. Tie sú následne smerované na ovládače portov bežiacie v režime jadra.⁵

³ napr. I/O rozhrania, medzi procesorové komunikačné mechanizmy, špecifiká kontrolórov prerušení, atď.

⁴ 16-bitový subsystém nie je prítomný v 64-bitových edíciách OS Windows a preto na nich nie je možné spúšťať aplikácie MS-DOS

⁵ ovládač paralelného portu alebo ovládač USB portu pre tlačiareň

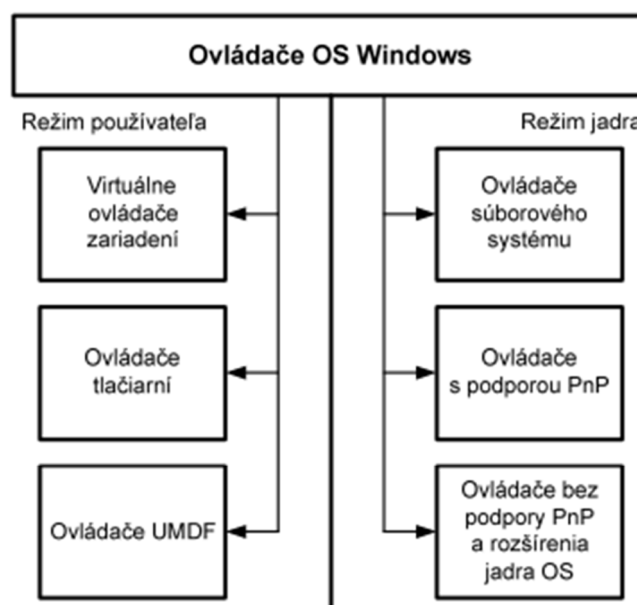
- **ovládače architektúry UMDF (rozhranie pre tvorbu ovládačov v režime používateľa) (angl. „UMDF drivers“)** - ovládač HW zariadenia bežiaci v režime používateľa. Komunikácia s podpornou knižnicou UMDF bežiacou v režime jadra je realizovaná pomocou ALPC (pokročilé lokálne volanie procedúr).

Pokročilé lokálne volanie procedúr (angl. „Advanced local procedure call“, ALPC) - forma komunikácie medzi procesmi pre vysokorýchlostné odovzdávanie správ, prístupná iba komponentom OS Windows, nie cez Windows API. Nahrádza starší spôsob LPC (lokálne volanie procedúr) (angl. „Local procedure call“) a má oproti nemu lepšiu škálovateľnosť, čo ju umožňuje použiť pre UMDF [1].

Ovládače v režime jadra:

- **ovládače súborového systému (angl. „file system drivers“)** - prijímajú I/O požiadavky na úrovni práce so súbormi a uspokojujú ich vysielaním špecifickejších požiadaviek ovládačom dátových úložísk alebo sieťových zariadení.
- **PnP (pripoj a používaj) ovládače (angl. „PnP drivers“)** - obsluhujú HW a spolupracujú so správcom napájania OS Windows a správcom PnP. Medzi tieto ovládače patria napr. ovládače dátových úložísk, vstupných zariadení a ovládače grafických a sieťových kariet.
- **ovládače nepodporujúce PnP (angl. „Non-PnP drivers“) a rozšírenia jadra (angl. „kernel extensions“)** - jedná sa o ovládače alebo moduly rozširujúce funkcionality OS. Nie sú prepojené so správcom napájania alebo správcom PnP, pretože väčšinou nespravujú skutočné fyzické zariadenie. Takýmto ovládačom je napr. ovládač sieťového zásobníka OS.

Ovládače v režime jadra sa ďalej delia na skupiny podľa využívaného modelu ovládača a úlohy ovládača pri práci so zariadením. Rozdelenie ovládačov v OS Windows je znázornené na obr. 2.2.



Obr. 2.2 - Rozdelenie typov ovládačov OS Windows

Pod modelom ovládača sa rozumie množina rozhraní, ktoré môže ovládač využívať. Ovládač, navrhnutý aby používal konkrétny model, musí používať rozhrania tohto modelu.

WDK podporuje tieto modely ovládačov:

- Windows Driver Model (WDM) (model ovládačov Windows);
- Kernel-Mode Driver Framework (KMDF) (rozhranie pre tvorbu ovládačov bežiacich v režime jadra);
- User-Mode Driver Framework (UMDF);
- Modely špecifické pre konkrétny typ zariadenia.

Ak rozhrania tvorby ovládačov v režime používateľa poskytujú možnosti potrebné pre ovládač, tak je výhodnejšie vyvíjať ovládače pomocou týchto rozhraní. Ďalšou voľbou je KMDF. Ak zariadenie vyžaduje špeciálny model nepodporovaný KMDF, je nutné použiť tento špeciálny model. V poslednom rade, ak nemôže byť použitý žiadny iný model, je nutné použiť WDM.

2.2.1 Windows driver model

Väčšina ovládačov Windows sú ovládače pre režim jadra zodpovedajúce špecifikácii WDM. Tento model umožňuje priamu komunikáciu ovládača s komponentmi Windows v režime jadra, napr. správa I/O operácií alebo správa podpory PnP. Umožňuje tvorbu ovládačov zariadení kompatibilných so všetkými verziami OS Windows na úrovni zdrojového kódu.

2.2.2 Kernel-Mode Driver Framework

KMDF je knižnica použiteľná pre tvorbu WDM ovládačov poskytujúca množinu rozhraní ľahšie použiteľných v porovnaní s WDM rozhraniami, riadi množstvo operácií, ktoré by museli byť inak spracovávané samotným ovládačom, a tiež zjednodušuje tvorbu ovládačov pre viacprocesorové systémy, pretože poskytuje množstvo z potrebnej synchronizácie.

Ovládačom je poskytované objektovo orientované rozhranie, nekomunikujú s komponentmi Windows v režime jadra priamo, ale pomocou metód KMDF objektov. Implementácia KMDF objektov využíva rozhrania WDM. KMDF zachytáva všetky I/O požiadavky zasielané ovládaču, niektoré spracováva, napr. PnP a PM (správa napájania), ostatné prevádza na spracovanie ovládaču.

2.2.3 User-Mode Driver Framework

UMDF je knižnica umožňujúca písanie ovládačov pracujúcich v režime používateľa. Táto možnosť je podporovaná iba pre niektoré typy zariadení a pre tieto zariadenia je preferovaná pred KMDF. UMDF poskytuje zjednotený model pracujúci na širokej škále tried zariadení, integruje inštaláciu a správu zariadení so štandardnými prostriedkami OS, ako napr. PnP a správa napájania. UMDF je navrhnutý pre podporu multimediálnych tried

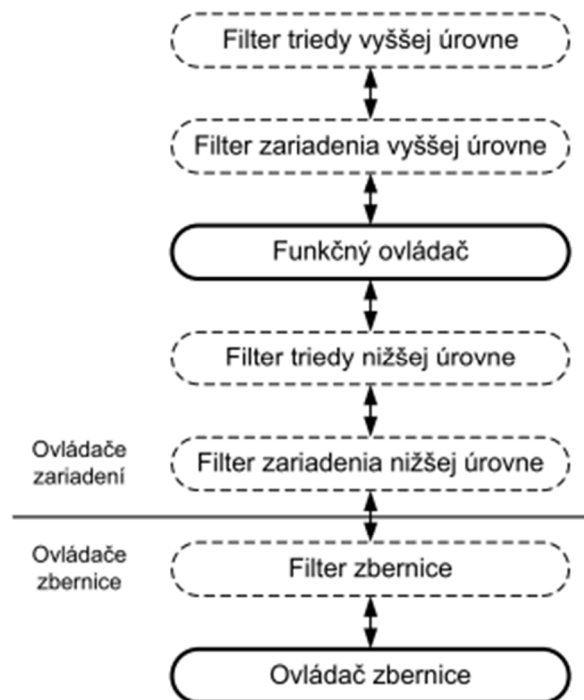
zariadení,⁶ kde presun ovládača z režimu jadra do režimu používateľa nie len výrazne zjednoduší tvorbu a ovládača a ovládač samotný, ale má aj pozitívny vplyv na stabilitu OS.

2.2.4 Modely špecifické pre konkrétny typ zariadenia

Pre ovládače niektorých typov zariadení musia byť použité modely špecifické pre konkrétny typ zariadenia. Takéto ovládače komunikujú s ovládačmi triedy zariadení, pomocou ktorých sú im poskytované špecifické rozhrania.

2.3 Vrstvová architektúra WDM

Ovládače modelu WDM podporujú spoluprácu so správcom napájania OS Windows, správcom PnP a taktiež prácu s technológiou WMI (výstroj správy Windows). Hierarchickú štruktúru WDM a radenie jednotlivých ovládačov zobrazuje obr. 2.3. Povinne musí byť prítomný iba ovládač zbernice a funkčný ovládač.⁷ Filtre môžu, ale nemusia byť prítomné.



Obr. 2.3 - Vrstvová architektúra WDM

WDM definuje tri typy ovládačov:

- ovládače zberníc (angl. „bus drivers“);⁸
- funkčné ovládače (angl. „function drivers“);
- filtre (angl. „filter drivers“).

⁶ napr. fotoaparáty, prehrávače hudby, atď.

⁷ funkcionality funkčného ovládača môže byť v niektorých prípadoch realizovaná ovládačom zbernice, jedná sa o jednoduchšie zariadenia, ktoré sú logicky zviazané so zbernicou [3]

⁸ Zbernicou je myslené ľubovoľné zariadenie, ku ktorému sa pripájajú iné fyzické, logické alebo virtuálne zariadenia. Zbernica môže byť klasická (PCI (prepojenie komponent periférií), SCSI (systémové rozhranie malých počítačov)), alebo sériový, paralelný, či USB (univerzálna sériová zbernica) port.

2.3.1 Ovládače zbernic

Slúžia na ovládanie individuálneho I/O zbernicového zariadenia a poskytujú funkcionality nezávislú na pripojenom zariadení každému slotu zbernice. Tieto ovládače majú tiež za úlohu zisťovať zariadenia pripojené na zbernicu a ohlasovať ich operačnému systému. Ovládač zbernice obsluhuje kontrolór zbernice (angl. „bus controller“), adaptér alebo most. OS Windows poskytuje ovládače pre všetky bežné typy zbernic,⁹ napr. PCI, PnP ISA (priemyselne štandardná architektúra), SCSI, USB. Iné ovládače zbernic môžu byť poskytované IHV (nezávislý tvorca zariadení). Pre každý typ zbernice v počítači musí byť v OS nutne nainštalovaný jeden ovládač zbernice. Ovládač zbernice môže obsluhovať viac zbernic rovnakého typu, ak sa v počítači nachádzajú. Medzi základné úlohy ovládača zbernice patrí zisťovanie pripojených zariadení, reakcia na výzvy PnP a PM, viacnásobný prístup k zbernici¹⁰ a správa zariadení pripojených na zbernicu.

2.3.2 Funkčné ovládače

Funkčné ovládače obsluhujú jednotlivé zariadenia. Jedná sa o hlavné ovládače z pohľadu zariadenia a pre použitie zariadenia sú vyžadované, ak zariadenie nie je používané v tzv. základnom režime (angl. „raw mode“). Základný režim je špeciálny režim funkcie zariadenia, kedy je zariadenie obsluhované ovládačom zbernice a nie funkčným ovládačom. Je možné tiež používať filtre zbernice.¹¹

Každé zariadenie môže byť obsluhované len jedným funkčným ovládačom, jeden funkčný ovládač ale môže naraz pracovať s viacerými zariadeniami. Funkčný ovládač poskytuje pracovné rozhranie zariadenia, typicky I/O operácie a správu napájania. Funkčné ovládače sú zvyčajne dodávané tvorcami zariadení a môžu byť realizované ako pár ovládač + mini ovládač, kde mini ovládač využíva rozhranie ovládača.

2.3.3 Filtre

Slúžia na filtrovanie I/O požiadaviek na zariadenie, triedu zariadení alebo zbernicu. Filtre sú voliteľné ovládače, ktoré pridávajú funkcionality alebo menia chovanie zariadenia. Každým filtrom môže byť obsluhovaných viac zariadení a počet filtrov pre jedno zariadenie v žiadnej z vrstiev nie je obmedzený.

Filtre sa rozdeľujú:

- **filtre zbernice (angl. „bus filter drivers“)** - typicky pridávajú funkcionality zbernici, napr. realizáciou proprietárnych rozšírení štandardizovanej zbernice. Príkladom môže byť ACPI (pokročilé rozhranie nastavení a správy napájania) filter, ktorý je vkladáný správcom

⁹ podpora konkrétnych typov zbernic ale závisí na verzii jadra OS Windows

¹⁰ v prípade, že ho zbernica podporuje

¹¹ Ohľadom použitia filtrov v základnom režime si literatúra [3] protirečí, keď je uvádzané, že je možné používať len filtre zbernice, ale na inom mieste, že je možné používať filtre všetkých úrovní. V ďalších zdrojoch, napr. [2] je spomínaná len možnosť využitia filtrov zbernice.

napájania nad ovládač zbernice každého ACPI kompatibilného zariadenia. Tento filter následne zodpovedá za správu napájania zariadení a pre ovládače na vyšších vrstvách je úplne transparentný.

- **filtre nižšej úrovne (angl. „lower-level filter drivers“)** - typicky upravujú správanie zariadení, napr. tak, aby zodpovedalo špecifikácii. Filtre nižšej úrovne monitorujú a prípadne menia I/O požiadavky na zariadenie (filtre zariadení nižšej úrovne, angl. „lower-level device filters“), alebo triedu zariadení (filtre tried nižšej úrovne, angl. „lower-level class filters“). Filtre nižšej úrovne sú hierarchicky zaradené pred funkčný ovládač zariadenia.
- **filtre vyššej úrovne (angl. „upper-level filter drivers“)** - typicky poskytujú zariadeniu funkcie s pridanou hodnotou. Filtre vyššej úrovne poskytujú služby jednotlivým zariadeniam (filtre zariadení vyššej úrovne, angl. „upper-level device filters“) alebo celým triedam zariadení (filtre tried vyššej úrovne, angl. „upper-level class filters“). Filtre vyššej úrovne pracujú nad funkčným ovládačom zariadenia.

2.3.4 Triedy zariadení OS Windows

Microsoft Windows definuje dve triedy zariadení:

- **trieda nastavenia zariadenia (angl. „device setup class“)** - popisuje skupinu zariadení, ktoré sú nastavované a konfigurované rovnakým spôsobom. Triedy nastavenia zariadenia sú zavedené pre uľahčenie inštalácie zariadení. Zariadenia, ktoré sú nastavované a konfigurované rovnakým spôsobom, sú zoskupené do tried, ktoré definujú postup inštalácie zariadenia. V systéme sú definované triedy nastavenia pre väčšinu zariadení, pričom ak žiadna z týchto tried nie je pre konkrétne zariadenie použiteľná, tak IHV môže definovať novú triedu.
- **trieda rozhrania zariadenia (angl. „device interface class“)** - popisuje skupinu vlastností, ktoré zariadenia poskytujú bez ohľadu na zbernicu, cez ktorú sú pripojené. Triedy rozhrania definujú prostriedky, ktorými ovládač sprístupňuje zariadenie aplikáciám a iným ovládačom. Každý ovládač zariadenia, s ktorým môžu spolupracovať prostriedky (ovládače, aplikácie) v režime používateľa, musí poskytovať určitý názov, podľa ktorého je zariadenie identifikované. Počínajúc od Windows 2000¹² sú na to využívané triedy rozhrania, do ktorých sa ovládače registrujú. Typicky sa ovládače zaregistrujú iba do jednej triedy rozhrania, ale ak má zariadenie funkcie nad rámec štandardných pre danú triedu, môže sa zaregistrovať aj do ďalších tried. Aplikácia využívajúca rozhranie si môže vyžiadať zoznam inštancií rozhrania a tak získať zariadenia, ktoré konkrétnu triedu rozhrania podporujú.

¹² verzia jadra OS 5.0 a vyššia

2.4 Štruktúra ovládača OS Windows

Keďže kód ovládača je vykonávaný I/O systémom OS, ovládač má predpísanú štruktúru, čo znamená, že musí implementovať funkcie, ktoré budú volané operačným systémom. Ovládače zariadení pozostávajú z niekoľkých procedúr volaných pre spracovanie rôznych častí I/O požiadaviek.

Nie všetky z nasledujúcich procedúr sú prítomné vo všetkých ovládačoch, niektoré sú voliteľné a niektoré majú zmysel iba pre určité druhy ovládačov. Značná časť týchto procedúr je nutná a aplikovateľná len pre ovládače fyzických zariadení.

Štartovacia procedúra (initialization routine) - vykonaná správcom I/O pri nahraní ovládača do OS. Má za úlohu naplniť systémové dátové štruktúry pre registráciu zvyšku procedúr v I/O správcovi a vykonáva potrebné globálne iniciácie. Je obdobou konštruktora z terminológie objektovo orientovaného programovania.

Procedúra pre pridanie zariadenia (add device routine) - ovládač podporujúci PnP obsahuje takúto procedúru. Správca PnP OS zasiela cez túto procedúru ovládaču oznámenie o nájdení nového zariadenia vždy, keď je detekované zariadenie, za ktoré je ovládač zodpovedný. Táto procedúra má za úlohu alokáciu objektu zariadenia prezentujúci zariadenie.

Odbavovacie procedúry (set of dispatch routines) - najdôležitejšie funkcie poskytované ovládačom zariadenia. Umožňujú napr. začatie komunikácie so zariadeniami (angl. „open“), ukončenie komunikácie so zariadením (angl. „close“), čítanie zo zariadenia (angl. „read“), zápis na zariadenie (angl. „write“) a použitie ľubovoľných iných schopností zariadenia, súborového systému či sieťového zariadenia. Keď je správca I/O volaný kvôli uskutočneniu I/O operácie, tak vygeneruje zodpovedajúci IRP (paket požiadavky prerušenia) paket a volá niektorú z odbavovacích procedúr ovládača zariadenia.

Procedúra pre začatie I/O operácie (start I/O routine) - ovládač môže použiť túto procedúru pre začatie presunu dát do zariadenia, alebo zo zariadenia. Procedúra je definovaná len v ovládačoch, ktoré sa v otázke radenia prichádzajúcich I/O požiadaviek spoliehajú na správcu I/O. Správca I/O zaraďuje požiadavky na ovládač za sebou a tak zabezpečuje, aby ovládač spracovával vždy len jeden IRP paket. Väčšina ovládačov môže spracovávať viac IRP paketov paralelne, sériové spracovanie ale má zmysel pre určité ovládače, napr. ovládač klávesnice.

Procedúra pre ošetrovanie prerušenia (interrupt service routine, ISR) - keď zariadenie vyvolá prerušenie, tak správca prerušení v jadre OS odovzdá riadenie tejto procedúre. V I/O modeli Windows ISR bežia s oprávneniami DIRQL (stupeň požiadavky prerušenia zariadenia), preto vykonávajú tak malé množstvo práce, ako to len ide, aby zbytočne neblokovali prerušenia s nižšou prioritou. ISR preto naplánuje volanie DPC (odložené volanie procedúr), ktoré beží na nižšom IRQL (stupeň požiadavky prerušenia), aby vykonalo zvyšok spracovania prerušenia. ISR disponujú len ovládače pre zariadenia riadené pomocou prerušení.

Procedúra pre spracovanie prerušenia (interrupt servicing DPC routine) - po ukončení ISR vykonáva väčšinu práce v spojitosti so spracovaním prerušenia zariadenia. Táto procedúra

beží na nižšom IRQL, aby neblokovala spracovanie prerušení s nižšou prioritou. Procedúra iniciuje dokončenie I/O operácie a začne ďalšiu naplánovanú I/O operáciu.

Procedúry pre dokončenie I/O operácií (I/O completion routines) - vrstvomý ovládač ich môže obsahovať, aby ho upozornili na dokončenie spracovania IRP ovládačom na nižšej vrstve. Príkladom môže byť volanie procedúry pre dokončenie I/O operácie ovládača súborového systému po ukončení prenosu dát ovládačom dátového úložiska. Ovládač je takto informovaný o výsledku operácie na nižšej vrstve a môže sa podľa neho rozhodnúť o ďalšej operácii.

Procedúry pre zrušenie I/O operácií (cancel I/O routine) - môžu byť definované, pokiaľ môže byť I/O operácia zrušená. Ak ovládač dostane IRP požiadavku na I/O operáciu, ktorá môže byť zrušená, priradí IRP rušiacu procedúru. Tá je zavolaná ak vlákno zadávajúce I/O požiadavku zanikne pred jej dokončením, alebo požiadavku zruší. Rušiaci procedúra je zodpovedná za uvoľnenie prostriedkov alokovaných pre spracovanie IRP a pre ukončenie operácie a vrátenie informácie o zrušení operácie vyšším vrstvám.

Procedúra pre rýchle odbavenie (fast dispatch procedure) - túto procedúru typicky poskytujú ovládače využívajúce služby správcu vyrovnávacej pamäte OS Windows, aby umožnili jadru obísť klasickú cestu spracovania I/O požiadaviek pri prístupe k ovládaču. I/O operácie ako napr. čítanie alebo zápis môžu byť vykonané rýchlo prístupom k vyrovnávacej pamäti namiesto obvyklej cesty, ktorou ide správca I/O, generujúcej nespojitú I/O operácie. Procedúry pre rýchle odbavenie sú tiež používané ako mechanizmy pre spätné volanie ovládačov súborového systému zo správcu pamäte (angl. „memory manager“) a správcu vyrovnávacej pamäte (angl. „cache manager“) [1].

Odstraňovacia procedúra (unload routine) - uvoľňuje všetky systémové zdroje používané ovládačom, aby mohol byť ovládač odstránený z pamäte. Zvyčajne sú tu uvoľnené zdroje alokovaná počas štartovacej procedúry. Ak to ovládač umožňuje, tak môže byť načítaný a odstránený počas behu systému, ale odstránenie je vždy vykonané až po uzavretí všetkých manipulátorov (angl. „handle“) zariadenia.

Procedúra pre oznámenie vypnutia systému (system shutdown notification routine) - táto procedúra umožňuje uvoľnenie prostriedkov v prípade vypínania systému.

Procedúry pre záznam chýb (error-logging routines) - tieto procedúry slúžia na zistenie výskytu neočakávaných chýb a oznámenie tejto skutočnosti správcovi I/O. Ten potom vloží záznam do denníka chýb.

2.5 Sieťové ovládače

Sieťové ovládače je možné rozdeliť:

- **ovládač sieťového adaptéru (angl. „network interface driver“)** - ovládač určený pre podporu konkrétnej sieťovej karty. Prekladá všeobecné povely od sieťového zásobníka na príkazy, ktoré realizuje sieťový hardware. Pre tvorbu ovládačov pre sieťové adaptéry sa využíva technológia NDIS (špecifikácia rozhrania sieťových ovládačov).

- **ovládač filtrujúci sieťovú premávku (sieťový filter, angl. „network filter“)** - ovládač pracujúci s dátovým tokom, či už prichádzajúcim do systému alebo v systéme vznikajúcim. Služi na úpravu vlastností dátového toku, napr. zabezpečenie QoS (kvalita služieb) či tvarovanie sieťovej premávky (angl. „traffic shaping“), alebo analýzu sieťovej premávky a reakciu na jej stav, napr. Firewall, IDS (systém detekcie prieniku) a IPS (systém zabránenia prieniku) systémy, ale aj napr. smerovanie sieťovej premávky.

2.5.1 Rozdiely medzi systémami NT 5.x a NT 6.x

Rozdiely v tvorbe ovládačov sieťových rozhraní vyplývajú najmä z toho, že OS Windows verzie jadra 6.x používa novú verziu NDIS. Tá je prispôbená zmenám v architektúre systému. Ovládače sieťových rozhraní vytvorené s použitím staršej verzie NDIS sú podporované naďalej, nemôžu ale využívať nové vlastnosti sieťového zásobníka.

OS Windows verzie jadra 6.x má nový sieťový zásobník, tzv. sieťový zásobník novej generácie (angl. „next generation network stack“), integrujúci sieťový protokol IPv4 (internetový protokol verzie 4) a IPv6 (internetový protokol verzie 6). Spôsoby priameho prístupu k sieťovému zásobníku z OS Windows verzie jadra 5.x nefungujú a tak niektoré staršie technológie, ako napr. ovládače zachycujúcich filtrov (angl. „hook driver“) alebo TDI (rozhranie prenosového ovládača) filtre sú považované za zastarané a buď ich nemožno, kvôli zmenám v architektúre sieťového subsystému, použiť vôbec alebo sa ich použitie neodporúča. Tieto staršie technológie sú ale nahradené novými technológiami, špeciálne vytvorenými a optimalizovanými pre konkrétne, často realizované, úlohy.¹³ Zmeny v technológiách pre spracovanie sieťovej premávky sú popísané v tab. 2.1.

Úprava produktu pre použitie s novou verziou sieťového zásobníka môže priniesť nové možnosti, ktoré neexistovali v predchádzajúcich verziách OS Windows, výrazne zjednodušiť sieťový ovládač a tiež zabezpečiť vyšší výkon.

Výrazný rozdiel spôsobuje to, že v OS Windows verzie jadra 6.x je pre automatickú inštaláciu ovládača správcom PnP nutný digitálny podpis balíčka ovládača. V prípade 64-bitových systémov je nutný digitálne podpísaný ovládač aj pre jeho zavedenie v prípade, že beží v režime jadra. Na takýchto systémoch je teda potrebné podpisovať ovládače už počas ich vývoja a testovania. Na OS Windows verzie jadra 5.x neboli pre ovládače vyžadované digitálne podpisy.

Tab. 2.1 - Zmeny v metódach spracovania sieťovej premávky medzi verziou jadra 5.x a 6.x

OS Windows verzie jadra 5.x	OS Windows verzie jadra 6.x
zachycujúci ovládač pre filtrovanie / pre Firewall pre jednoduché spracovanie paketov	aplikácia alebo služba v režime používateľa využívajúca API WFP

¹³ napr. ovládače zachycujúcich filtrov boli nahradené ovládačmi Windows Filtering Platform

OS Windows verzie jadra 5.x	OS Windows verzie jadra 6.x
zachycujúci ovládač pre filtrovanie / pre Firewall pre hĺbkovú analýzu alebo úpravu paketov	ovládač sieťovej alebo transportnej vrstvy a prípadná aplikácia v režime používateľa využívajúca API WFP
ovládač TDI pre filtrovanie použitý pre jednoduché filtrovanie premávky	aplikácia alebo služba v režime používateľa využívajúca API WFP
ovládač TDI pre filtrovanie použitý pre hĺbkovú analýzu alebo modifikáciu	ovládač na aplikačnej vrstve a prípadná aplikácia v režime používateľa využívajúca API WFP
ovládač TDI pre filtrovanie použitý pre správu sieťových spojení a dátových tokov	ovládač WFP a prípadná aplikácia v režime používateľa využívajúca API WFP, pre pokročilú správu sieťových spojení ovládač TDI pre filtrovanie
Winsock LSP (poskytovateľ vrstvených služieb) filter	aplikácia alebo služba v režime používateľa využívajúca API WFP
Winsock LSP filter pre hĺbkovú analýzu alebo modifikáciu	ovládač sieťovej alebo transportnej vrstvy a prípadná aplikácia v režime používateľa využívajúca API WFP
medziľahlý NDIS ovládač pre jednoduché filtrovanie premávky	aplikácia alebo služba v režime používateľa využívajúca API WFP
medziľahlý NDIS ovládač pre správu sieťových spojení a dátových tokov	ovládač na aplikačnej vrstve a prípadná aplikácia v režime používateľa využívajúca API WFP
NDIS ovládač filtrujúci podľa adresy linkovej vrstvy	NDIS ovládač pre filtrovanie podľa adresy linkovej vrstvy, WFP nepodporuje filtrovanie podľa adresy linkovej vrstvy

2.6 Filtrovanie sieťovej premávky

Pre filtrovanie sieťovej premávky existuje niekoľko možností. Celé filtrovanie je možné vykonať v režime používateľa, bez potreby písania ovládačov. Tento spôsob ale má množstvo obmedzení a pre bezpečnostné aplikácie je úplne nepoužiteľný.¹⁴ Pre účely manipulácie záhlavia IP paketov by však použitý byť mohol, pokiaľ neprekáža výkonová degradácia spôsobená presúvaním dát z režimu jadra do režimu používateľa a naspäť.

Metódy pre filtrovanie sieťovej premávky v režime používateľa:

- **vrstvový poskytovateľ služieb Winsock (angl. „Winsock LSP“)** - pre tento spôsob je využívaná technológia Winsock SPI, ktorá je dobre zdokumentovaná v MSDN (sieť vývojárov spoločnosti Microsoft) a periodikách spoločnosti Microsoft, napr. [9], taktiež k nej existuje dostatočné množstvo vhodných príkladov. Metóda je použiteľná napr. na zabezpečenie QoS alebo manipuláciu s dátovým tokom, nie je vhodná pre bezpečnostné

¹⁴ nežiaduci kód (vírus, trójsky kôň, atď.) môže riešenia v režime používateľa jednoducho obísť použitím jednoduchého ovládača v režime jadra, napr. TDI ovládač

aplikácie alebo pre filtrovanie na smerovači¹⁵ a manipuláciu s hlavičkami jednotlivých vrstiev.

- **rozhranie Windows 2000 pre filtráciu paketov (angl. „Windows 2000 packet filtering interface“)** - API poskytované Windows od verzie 2000 (jadro 5.0), pomocou ktorého môže aplikácia v režime používateľa zaviesť sadu filtrov, ktoré budú sieťovým zásobníkom OS používané pre rozhodnutie o zahodení paketu. Ďalej pravidlá je možné založiť len na IP adresách odosielateľa či príjemcu alebo na porte služby. Táto metóda je preto vhodná len pre jednoduchý Firewall.

Možnosti pre filtrovanie sieťovej premávky v režime jadra:

- **filter Winsock v režime jadra (angl. „kernel-mode sockets filter“)** - táto metóda je založená na zachytávaní komunikácie medzi súčastami Winsock v režime používateľa a súčastami Winsock v režime jadra. Reálne možnosti sú však prakticky rovnaké ako v prípade metódy Winsock LSP a toto rozhranie sa do značnej miery mení v podstate v každej verzii OS Windows, čo je prekážkou v portabilite kódu.
- **filter pre rozhranie ovládača transportnej vrstvy (angl. „TDI filter driver“)** - táto metóda spočíva v zachytávaní všetkých volaní objektov sieťového zásobníka. Je použitá v mnohých komerčných bezpečnostných riešeniach typu Firewall pre osobné použitie, ale na pokročilejšie bezpečnostné aplikácie nie je vhodná, keďže pracuje až nad sieťovým zásobníkom. Zároveň sa jedná o riešenie zložité, ktoré sa pre aplikácie kde existuje iné riešenie neodporúča používať.
- **medziľahlý ovládač NDIS (angl. „NDIS intermediate driver“)** - tento typ ovládača je vhodný najmä pre bezpečnostné aplikácie, ale má zložitú inštaláciu a práca s nimi je pre koncového používateľa pomerne zložitá. Tieto ovládače musia byť digitálne podpísané.
- **ovládač zachycujúceho filtra Windows (angl. „Windows filter-hook driver“)** - metóda použiteľná na Windows verzie jadra 5.x. Na vyšších verziách nie je API tejto technológie vôbec prístupné a takéto ovládače teda nie je možné použiť. Technológia bola na Windows verzie jadra 6.x nahradená Windows Filtering Platform.
- **ovládač zachycujúceho filtra NDIS (angl. „NDIS hooking filter driver“)** - táto metóda je založená na zachytávaní podskupiny funkcií NDIS, ktoré umožňujú sledovať registráciu protokolov nainštalovaných v OS a sieťové rozhrania, ktoré otvárajú. Výhodou metódy je jednoduchá inštalácia ovládačov a priama podpora sieťových technológií Windows bez väčšej dodatočnej práce. Nevýhodou je, že táto technológia neposkytuje jednoduché API pre manipuláciu dátových jednotiek. Využitie tejto technológie by vyžadovalo tvorbu veľkého množstva podporného kódu.
- **ovládač WFP (angl. „WFP driver“)** - metóda použiteľná na Windows verzie jadra 6.x, vytvorená špeciálne pre manipuláciu dátových jednotiek a tokov na všetkých vrstvách

¹⁵ smerovanie je realizované na sieťovej vrstve, v niektorých prípadoch na MAC podvrstve linkovej vrstvy, zatiaľ čo Winsock LSP je záležitosť relačnej vrstvy

sieťového zásobníka. Jedná sa o pomerne mladú technológiu, ktorá navyše trpela veľmi závažnými chybami. Je ale špeciálne určená pre bezpečnostné aplikácie, analýzu a úpravu paketov a sú na nej založené aj systémové komponenty OS Windows, ako napr. preklad sieťových adries (NAT), zabezpečenie sieťového protokolu IP (IPSec) či Windows Firewall.

2.6.1 Transport driver interface

Klienti TDI sú staršie (angl. „legacy“) ovládače v režime jadra zvyčajne implementujúce časť sieťového API bežiacu v režime jadra. Názov vyplýva z toho, že IRP pakety nimi zasielané protokolovému zásobníku sú formátované podľa štandardu TDI, popísaného v literatúre [3]. Tento štandard špecifikuje programovacie rozhranie pre ovládače zariadení bežiace v režime jadra.

Transportéri TDI sú protokolové ovládače v režime jadra. Prijímajú IRP pakety od klientov TDI a vybavujú požiadavky v nich formulované. Toto spracovanie môže vyžadovať sieťovú komunikáciu, pridávanie hlavičiek protokolov transportnej a sieťovej vrstvy či komunikáciu so sieťovým adaptérom pomocou NDIS funkcií. Transportéri TDI zabezpečujú sieťovú komunikáciu a odtieňujú TDI klientov od operácií ako segmentácia a spätné skladanie správ, zabezpečenie správneho poradia správ, potvrdzovanie správ a vyžiadanie opakovania v prípade neúspešného prenosu [1].

2.6.2 Network driver interface specification

Protokolový zásobník musí pre načítanie alebo zapísanie správy vo svojom protokolovom formáte použiť sieťový adaptér [1]. Keďže nie je možné, aby protokolový zásobník poznal špecifiká všetkých sieťových adaptérov,¹⁶ tvorcovia sieťových adaptérov pre ne poskytujú ovládače schopné spracovávať správy a prijímať či odosielať ich pomocou HW zariadenia.

V roku 1989 spoločnosti Microsoft a 3Com spoločne vyvinuli špecifikáciu NDIS, ktorá umožňuje sieťovému zásobníku komunikovať so sieťovým adaptérom spôsobom nezávislým na konkrétnom type použitého sieťového adaptéru. Od vtedy je špecifikácia NDIS stále vyvíjaná a nové verzie, podporujúce nové vlastnosti sieťových adaptérov, sú dodávané ako súčasť každej generácie OS Windows. Prehľad verzií NDIS v rôznych OS spoločnosti Microsoft je uvedený v tab. 2.2. Ovládače sieťových adaptérov zodpovedajúce špecifikácii NDIS sa označujú ako NDIS ovládače.

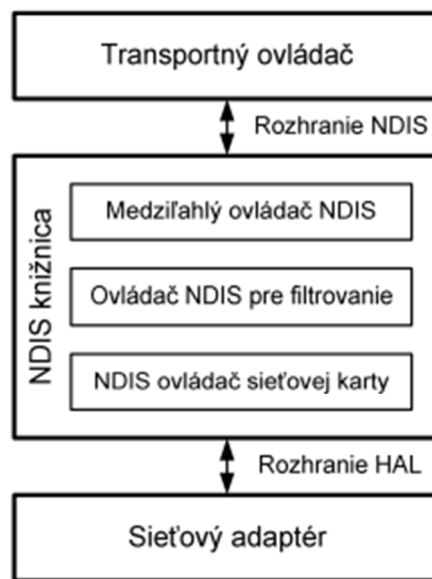
Podporná knižnica NDIS je hranicou medzi sieťovým zásobníkom a NDIS ovládačom a je využívaná sieťovým zásobníkom na formátovanie príkazov pre NDIS ovládač. NDIS ovládače s jej pomocou prijímajú žiadosti a zasielajú na ne odpovede. Podporná knižnica NDIS ale neslúži len ako poskytovateľ pomocných funkcií, poskytuje celé prostredie pre spúšťanie NDIS ovládačov. NDIS ovládače nie sú skutočnými ovládačmi Windows, pretože nemôžu fungovať bez tohto zapuzdrenia. NDIS ovládače taktiež nespracovávajú IRP pakety, sieťový

¹⁶ počet typov sieťových kariet na trhu dosahuje rádovo hodnotu 10^3 [4]

zásobník volá funkcie z pomocnej knižnice [1]. Rozhranie poskytované pomocnou knižnicou pre prácu so sieťovým adaptérom je priamo prekladané na zodpovedajúce funkcie HAL. Architektúra NDIS je zobrazená na obr. 2.4.

Tab. 2.2 - Verzie NDIS podporované v operačných systémoch

NDIS	Operačný systém
2.0	MS-DOS, Windows 3.1, OS/2
3.0	Windows for Workgroups 3.11, Windows NT 3.5
3.1	Windows 95
4.0	Windows 95 OSR2, Windows NT 4.0, Windows CE 3.0
4.1	Windows 98, Windows NT 4.0 SP3
5.0	Windows 98 SE, Windows Me, Windows 2000
5.1	Windows XP, Windows Server 2003, Windows CE 4.x, Windows CE 5.0
5.2	Windows Server 2003 SP2
6.0	Windows Vista
6.1	Windows Vista SP1, Windows Server 2008
6.20	Windows 7, Windows Server 2008 RC2



Obr. 2.4 - Architektúra NDIS

NDIS ovládač môže oznamovať aktivitu sieťového média, čo umožňuje informovať používateľa, ale tiež umožňuje sieťovému zásobníku a iným aplikáciám byť informovanými o aktuálnom stave a vhodne reagovať. Príkladom môže byť prehodnotenie aktuálnosti automaticky získaných informáciách o sieťovom adresovaní¹⁷ sieťovým zásobníkom.

Beh NDIS ovládačov môže byť pozastavený a znovu obnovený, čo umožňuje zmenu nastavení počas behu OS. Toto dovoľuje vykonávanie dynamických operácií, ako napr. registráciu sieťového zásobníka, bez reštartovania operačného systému.

¹⁷ získané od DHCP serveru

Deložovanie úloh umožňuje využívanie sieťovej karty na vykonávanie náročných operácií, ako napr. výpočet kontrolných súm. To odľahčuje procesor a má priaznivý vplyv na celkový výkon systému. Škálovateľnosť umožňuje vykonávať príjem paketov na viacerých procesoroch súčasne, podľa aktuálneho vyťaženia systému.

Technológia Wake-on-LAN (lokálna sieť) dovoľuje sieťovému adaptéru prebrať OS z režimu spánku. Pre signalizáciu k prebudeniu môže slúžiť niekoľko typov podnetov.

Možné podnety pre prebudenie OS z režimu spánku:

- pripojenie média do sieťového adaptéru;
- prijatie špeciálneho vzoru zaregistrovaného sieťovým zásobníkom;
- prijatie magického paketu.¹⁸

Rozdelenie hlavičky a dát rámcov do zvláštnych vyrovnávacích pamätí umožňuje sieťovým adaptérom s podporou tejto technológie zvýšiť výkon sieťového subsystému. Výhoda je, že rozdelené vyrovnávacie pamäte sa zmestia do menších oblastí pamäte, čo spôsobuje efektívnejšie využitie prostriedkov a rýchlejší prístup k hlavičkám paketov.

Spojovo orientované NDIS (CoNDIS, angl. „connection-oriented NDIS“) umožňuje NDIS ovládačom spravovať na linkovej vrstve spojovo orientované služby, ako napr. PPP (protokol spojenia z bodu do bodu) spojenia. Takéto NDIS ovládače používajú rovnaké API ako bežné NDIS ovládače, rozdiel je však v tom, že zasielajú pakety cez zavedené spojenie namiesto sieťového média [1].

NDIS model tiež podporuje zmiešané ovládače sieťovej a transportnej vrstvy, nazývané medziľahlé NDIS ovládače (angl. „NDIS intermediate driver“). Tieto ovládače sú umiestnené medzi transportné ovládače a NDIS ovládače, vďaka čomu nimi prechádza všetka sieťová premávka. Voči NDIS ovládaču sa medziľahlý ovládač správa ako transportný ovládač a voči transportnému ovládaču sa správa ako NDIS ovládač. Medziľahlé NDIS ovládače sa využívajú napr. pre systémy odolné voči chybám (angl. „fault tolerant“) alebo systémy s rozložením záťaže (angl. „load balancing“) [1].

NDIS model podporuje aj odľahčené ovládače pre filtrovanie (angl. „lightweight filter driver“), podobné medziľahlým NDIS ovládačom, ale navrhnuté špeciálne pre filtrovanie sieťovej premávky. Podporujú dynamické vkladanie a vyberanie počas behu, a keďže tieto filtre nie sú prepojené so žiadnym transportným ovládačom, majú možnosť spracovávať takmer všetku komunikáciu sieťového adaptéru.¹⁹ Je možné pre filtrovanie vybrať iba určité zariadenia a nechať premávku iných obchádzať filter. Tieto nastavenia je taktiež možné meniť počas behu.

¹⁸ paket obsahujúci 16 krát zopakovanú sieťovú (MAC) adresu adaptéru

¹⁹ okrem častí spracovaných priamo ovládačom sieťovej karty

2.7 Windows filtering platform

WFP je bohatá a rozšíriteľná platforma pre monitorovanie, zachytávanie a spracovávanie sieťovej premávky na všetkých úrovniach sieťového zásobníka [1]. Nahrádza rozhrania pre filtrovanie sieťovej premávky vo Windows XP a Windows Server 2003. WFP pozostáva z množiny spojov so sieťovým zásobníkom (angl. „network stack“) a filtračného jadra riadiaceho spoluprácu so sieťovým zásobníkom [6]. WFP poskytuje filtračnú infraštruktúru do ktorej môžu ISV zapájať špecializované filtračné moduly. Niektoré sieťové služby OS Windows rozširujú vlastnosti štandardného systémového protokolového ovládača TCP/IP (protokol s riedením vysielania / Internetový protokol) pomocou WFP [1].

WFP je množina API a služieb operačného systému poskytujúca platformu pre tvorbu aplikácií pre filtrovanie sieťovej premávky. API umožňuje vývojárom písať kód spolupracujúci so spracovaním paketov, ktoré sa deje na niekoľkých vrstvách sieťového zásobníka operačného systému. Sieťová premávka môže byť filtrovaná a tiež modifikovaná predtým, ako dosiahne svoj cieľ. WFP poskytuje jednoduchšiu vývojovú platformu a je navrhnuté, aby nahradilo predchádzajúce technológie pre filtrovanie paketov.²⁰ V OS Windows verzii jadra 6.x,²¹ kvôli novému sieťovému zásobníku, už zachycujúci ovládač pre Firewall a zachycujúci ovládač pre filtrovanie nie sú prístupné a namiesto nich je odporúčané využívať WFP [6].

WFP je vhodné na tvorbu aplikácií ako Firewall, IDS systémy, antivírusové programy, nástroje na sledovanie stavu siete alebo rodičovské zámky. Pre programy typu Firewall poskytuje podporu ako autentizovanú komunikáciu a dynamickú konfiguráciu založenú na správaní aplikácií. Taktiež sú poskytované služby pre správu postupov IPSec (angl. „IPSec management policy“), oznamovania zmien (angl. „change notification“), diagnostiku stavu siete a filtrovanie podľa stavu spojenia (angl. „stateful packet inspection“) [6].

WFP API sa skladá z API pre režim používateľa a API pre režim jadra. API pre režim používateľa je poskytované aplikáciám a API pre režim jadra je využívané ovládačmi.

2.7.1 Architektúra WFP

Prepojenie rôznych komponent WFP s protokolovým zásobníkom TCP/IP OS Windows je zobrazené na obr. 2.5.

Súčasťami WFP sú:

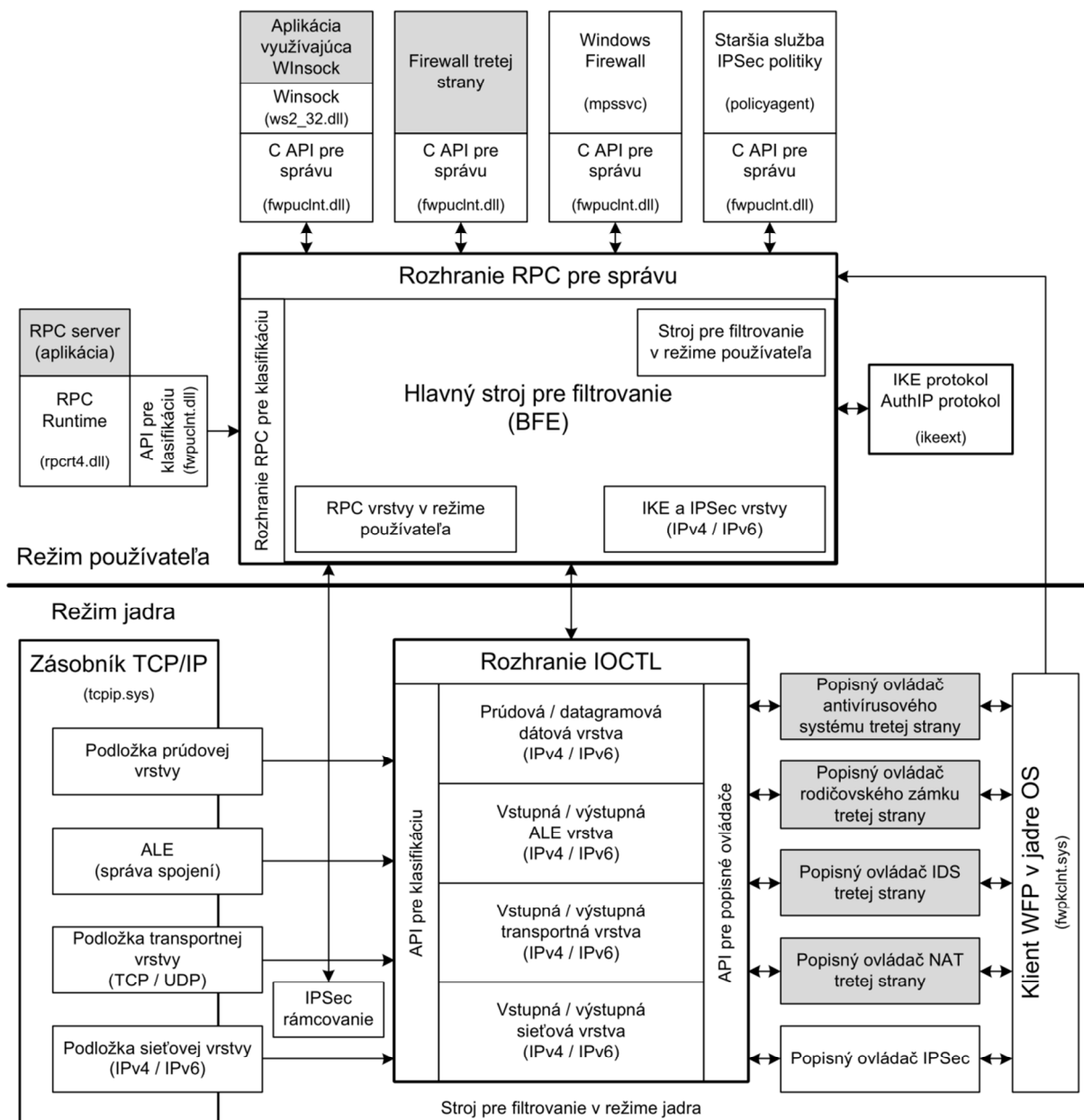
- **stroj pre filtrovanie (angl. „filtering engine“)** - je prítomný ako v režime používateľa, tak v režime jadra a vykonáva všetky filtračné úlohy na sieti. Každá súčasť stroja pre filtrovanie pozostáva z filtračných vrstiev, jednej pre každú vrstvu sieťového zásobníka. Stroj v režime používateľa, zodpovedný okrem iného za spracovávanie RPC (vzdialené volanie procedúr) a IPSec rámcovanie, obsahuje asi 10 filtrov, zatiaľ čo stroj v režime

²⁰ TDI filtre, NDIS filtre, poskytovatelia vrstvomých služieb Winsock LSP

²¹ Windows Vista, Windows Server 2008, Windows 7, Windows Server 2008 R2

jadra, vykonávajúci filtrovanie na sieťovej a transportnej vrstve sieťového zásobníka, ich obsahuje asi 50 [1].

- **podkladové moduly (angl. „shims“)** - súčasti v režime jadra umiestnené medzi sieťový zásobník a stroj pre filtrovanie. Sú zodpovedné za rozhodovanie, či bude sieťová premávka zablokovaná alebo prepustená, podľa správania, ktoré im definuje stroj pre filtrovanie. Podkladový modul najprv analyzuje prichádzajúce dáta a hľadá zodpovedajúce záznamy v tabuľke stoja pre filtrovanie, následne požiada stroj pre filtrovanie o rozhodnutie o opatreniach voči prichádzajúcim dátam a nakoniec tieto opatrenia vykoná.



Obr. 2.5 - Architektúra Windows Filtering Platform

- **hlavný stroj pre filtrovanie (angl. „base filtering engine“)** - služba OS v režime používateľa spravujúca všetky činnosti WFP. Je zodpovedná za pridávanie a odoberanie

filtrův zo zásobníka WFP, správu nastavení jednotlivých filtrův a uplatňovanie bezpečnostných zásad na databázu filtrův.

- **popisný ovládač (angl. „callout driver“)** - súčasť WFP pracujúce v režime jadra pridávajúce vlastnú filtračnú funkcionality nad rámec základnej podpory poskytovanej WFP. Popisný ovládač priradí popisné funkcie k jednej alebo viacerým filtračným vrstvám stoja pre filtrovanie v režime používateľa a WFP následne umožňuje týmto popisným funkciám vykonávať hĺbkovú analýzu paketův a taktiež ich modifikáciu.

2.8 Technológie tretích strán

Pre filtrovanie sieťovej premávky je možné okrem technológií spoločnosti Microsoft, zabudovaných priamo do OS Windows, možné použiť aj niekoľko produktův tretích strán.

2.8.1 CACE Technologies WinPcap

WinPcap je open-source implementácia knižnice libpcap, slúžiacej na zachytávanie a filtrovanie paketův a prácu s nimi aplikáciám v režime používateľa, pre architektúru Win32. Pôvodne bol WinPcap vyvíjaný na Turínskej technickej univerzite, neskôr, v roku 2005, vývoj prebrala spoločnosť CACE Technologies [15].

WinPcap obsahuje všetky špecifiká dobre známej knižnice libpcap z prostredia OS UNIX a umožňuje tak tvorbu aplikácií analyzujúcich zachytávanie paketův a analýzu sieťovej premávky portabilných na úrovni zdrojového kódu so systémami architektúry Windows a UNIX [15].

WinPcap obsahuje funkcie použiteľné pre tieto typy úloh:

- získanie zoznamu prístupných sieťových adaptérov;
- získanie informácií o konkrétnom adaptéri, napr. jeho popis a zoznam adres sieťovej vrstvy;
- zachytávanie paketův na jednej z prítomných sieťových kariet;
- zasielanie paketův do siete;
- efektívne ukladanie zachytených paketův do súboru a ich opätovné načítanie pomocou rozhrania podobného klasickému zachytávaniu;
- vytváranie vysokoúrovňových filtrův pre zachytávanie sieťovej premávky.

Výhodami tohto riešenia sú realizácia v režime používateľa, nulová cena, keďže sa jedná o riešenie s otvoreným zdrojovým kódom (angl. „open-source“) a prenositeľnosť na systémy architektúry UNIX na úrovni zdrojového kódu. Nevýhodou je nemožnosť pracovať so sieťovou premávkou pochádzajúcou zo systému, na ktorom aplikácia beží a degradácia výkonu kvôli nutnosti kopírovania dát z režimu jadra do režimu používateľa a naspäť.

2.8.2 NT Kernel WinpkFilter 3.0

Jedná sa o vysoko výkonnú štruktúru pre filtráciu paketov pre OS Windows architektúry Chicago²² a NT²³ s podporou aj pre 64-bitové systémy.²⁴ Umožňuje transparentné filtrovanie nespracovaných paketov (angl. „raw packet“), či už analýzu, alebo aj modifikáciu, s minimálnym vplyvom na sieťovú aktivitu [16]. Technológia je založená na vlastnom ovládači typu zachycujúci filter NDIS, takže ku všetkým možnostiam riešenia je možné pristupovať v režime používateľa a na programovanie je možné použiť prakticky ľubovoľný jazyk či vývojové prostredie.²⁵

Vývoj aplikácií pre režim používateľa je tiež výrazne jednoduchší v porovnaní s programovaním pre režim jadra. Ak sú však dôležitá maximalizácia výkonu vyvíjaného riešenia, je možné použiť aj dobre popísané API pre režim jadra. V tomto prípade nedôjde k 30-40% výkonovej degradácii kvôli presúvaniu dát z režimu jadra do režimu používateľa a následne po spracovaní naspäť [16]. Autor ale odporúča, ako jednoduchšiu alternatívu, úpravu kódu samotného ovládača.²⁶

Výhodou je možnosť riešiť problém programovaním v režime používateľa, čo ale prináša degradáciu výkonu. Nevýhodou je vysoká cena, keďže pre implementáciu ovládača podľa zadania je potrebný najvyšší variant licencie. Prehľad licencií je možné nájsť v tab. 2.3, pričom cena prechodu na vyššiu licenciu je rovná rozdielu cien novej licencie a pôvodnej licencie. Ďalšou nevýhodou je problém so spoluprácou viacerých aplikácií používajúcich túto technológiu na jednom systéme. Pre každú aplikáciu je potrebná upravená verzia ovládača aby nedochádzalo ku konfliktom.

Tab. 2.3 - Možnosti licencií NT Kernel WinpkFilter

Typ licencie	Distribúcia ovládačov	Zdrojový kód	Cena (1 rok) [€]	Predĺženie (1 rok) [€]
Individuálna	NIE	NIE	95	75
Vývojárska	ÁNO	NIE	1495	1195
So zdrojovým kódom	ÁNO	ÁNO	3495	2795

2.9 Praktické poznámky k tvorbe ovládačov

Pre tvorbu ovládača je vybraná technológia Windows Filtering Platform. Výhodou tejto technológie je, že je priamo určená na tvorbu systémov pre filtrovanie sieťovej premávky na všetkých úrovniach sieťového zásobníka pre OS Windows verzie jadra 6.x. Taktiež poskytuje dobrý výkon, pretože pri vykonávaní svojej funkcie ovládač nevyžaduje presuny dát a zmeny režimov behu medzi režimom jadra a režimom používateľa. Pre túto technológiu nie je nutné

²² všetky verzie Windows 95, Windows 98, Windows 98 SE, Windows ME

²³ všetky verzie Windows NT, Windows 2000, Windows XP, Windows 2003, Windows Vista, Windows 7, Windows 2008, Windows 2008 R2

²⁴ 64-bitové systémy vyžadujú digitálne podpísané ovládače, ktoré NT Kernel Resources neposkytujú, v prípade produkčného nasadenia na takýchto systémoch je preto potrebné podpísať ovládače samostatne [7]

²⁵ nutnou a dostačujúcou požiadavkou na jazyk je možnosť volať funkcie z DLL knižníc

²⁶ pre prístup k zdrojovým kódom je potrebná najvyššia verzia licencie v cene 3495€

zakúpenie žiadnej dodatočnej licencie a je vyriešená spolupráca s inými ovládačmi, či už dodávanými s OS alebo od tretích strán.

Po ukončení tvorby ovládača je nutné ho zostaviť, opatriť potrebným podpisom a inštalačnými informáciami a nakoniec distribuovať koncovému používateľovi.

2.9.1 Zostavenie ovládača

Zostavenie ovládača je vykonané v špeciálnom prostredí poskytovanom WDK (nástroj ovládačov Windows). Toto prostredie je podobné tomu, ktoré sa využíva na vnútorné zostavovanie ovládačov spoločnosťou Microsoft. Na zostavenie ovládača nie je možné použiť štandardné prostriedky na zostavovanie aplikácií, aké poskytuje napr. Microsoft Visual Studio, a preto WDK obsahuje vlastné nástroje:

- **build.exe** - nástroj na zostavovanie ovládačov, automaticky volá ostatné nástroje podľa použitých prepínačov a popisného súboru pre zostavenie (tzv. makefile);
- **nmake.exe** - nástroj pre čítanie popisných súborov pre zostavenie;
- **cl.exe** - kompilátor jazyka C;
- **link.exe** - spájací program (tzv. linker).

Pri zostavovaní ovládača sa postupuje nasledovne:

- **Spustenie prostredia na zostavenie ovládača** - V menu Štart sa postupuje touto cestou: Štart → All Programs / Všetky programy → Windows Driver Kits → (verzia WDK) → Build Environment → (operačný systém) → (prostredie na zostavenie).

Prostredia sú rozdelené podľa architektúry cieľového systému:

- **x86** - systém založený na 32-bitovom procesore Intel alebo inom, kompatibilnom procesore;
- **x64** - systém založený na 64-bitovom procesore AMD (pokročilé mikroskopické zariadenia) alebo inom, kompatibilnom procesore;
- **IA-64** - systém založený na 64-bitovej architektúre Intel Alpha (procesory Itanium).

A podľa vývojovej fázy ovládača:

- **voľné zostavenie (angl. „free build“)** - zostavený ovládač je určený pre produkčné použitie, kód ovládača je optimalizovaný a kód pre hľadanie chýb sa v ovládači nenachádza. Takýto ovládač sa tiež používa pre hodnotenie výkonu.
- **kontrolované zostavenie (angl. „checked build“)** - zostavený ovládač je určený pre použitie počas vývoja, obsahuje kód pre zjednodušenie hľadania chýb a kód ovládača nie je optimalizovaný.

Ovládače by mali byť najprv zostavené a vyskúšané ako kontrolované, a keď sa overí ich činnosť, tak by mali byť zostavené ako voľné a takto znovu odskúšané [3]. Po výbere jedného z prostredí je spustený špeciálny príkazový riadok, v ktorom je nastavené množstvo systémových premenných zjednodušujúcich prácu pri zostavovaní a skúšaní

ovládačov. Nástroj na zostavovanie ovládačov (build.exe) by mal byť vždy spúšťaný z tohto prostredia.

- **Zmena pracovného adresára** - následne je potrebné zmeniť pracovný adresár na adresár obsahujúci zdrojové súbory ovládača príkazom:

```
cd (adresár)
```

kde (adresár) je cesta v súborovom systéme k adresáru so zdrojovými súbormi ovládača. Prípadne je tiež možné použiť príkaz:

```
(jednotka):
```

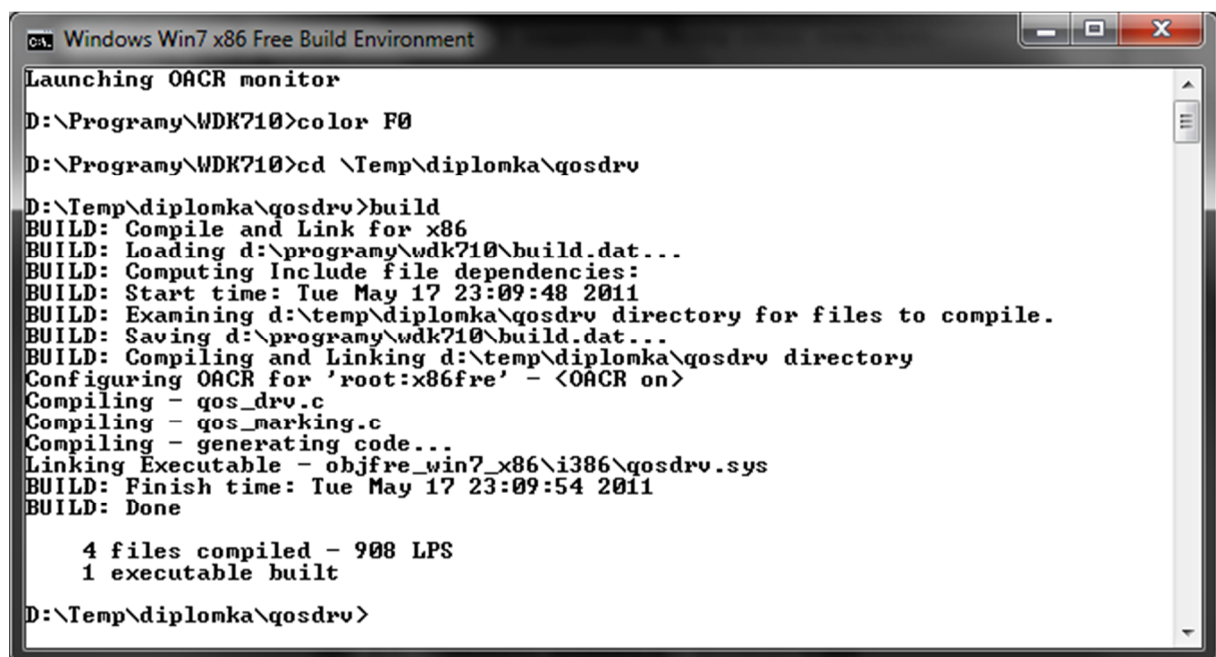
kde (jednotka) je jednopísmenové označenie diskovej jednotky, ak sa zdrojové súbory ovládača nachádzajú na inom logickom disku ako inštalačný adresár WDK.

- **Spustenie nástroja na zostavenie ovládača** - samotné zostavenie ovládača sa vykoná príkazom:

```
build
```

ktorý spustí program build.exe. Tento program postupuje podľa súboru pre zostavenie a podľa potreby volá ďalšie nástroje. Výstupom programu je samotný ovládač, ktorého cieľové určenie závisí na zvolenom prostredí.

Postup zostavenia ovládača je znázornený na obr. 2.6 spolu s hláseniami nástroja pre zostavenie.



```
Windows Win7 x86 Free Build Environment
Launching OACR monitor
D:\Programy\WDK710>color F0
D:\Programy\WDK710>cd \Temp\diplomka\qosdrv
D:\Temp\diplomka\qosdrv>build
BUILD: Compile and Link for x86
BUILD: Loading d:\programy\wdk710\build.dat...
BUILD: Computing Include file dependencies:
BUILD: Start time: Tue May 17 23:09:48 2011
BUILD: Examining d:\temp\diplomka\qosdrv directory for files to compile.
BUILD: Saving d:\programy\wdk710\build.dat...
BUILD: Compiling and Linking d:\temp\diplomka\qosdrv directory
Configuring OACR for 'root:x86fre' - <OACR on>
Compiling - qos_drv.c
Compiling - qos_marking.c
Compiling - generating code...
Linking Executable - objfre_win7_x86\i386\qosdrv.sys
BUILD: Finish time: Tue May 17 23:09:54 2011
BUILD: Done

    4 files compiled - 908 LPS
    1 executable built
D:\Temp\diplomka\qosdrv>
```

Obr. 2.6 - Voľné zostavenie ovládača pre Windows 7 x86

2.9.2 Inštalácia ovládača

Ovládače sú distribuované koncovým používateľom vo forme tzv. balíčkov (angl. „driver package“). Balíček pozostáva zo všetkých súčastí nutných k inštalácii zariadenia a jeho podpore v OS Windows. Pre inštaláciu zariadenia alebo ovládača sú nutné súčasti dodané

systemom aj tvorcom zariadenia alebo produktu. Operačný systém poskytuje všeobecné inštaláčne programy pre všetky triedy zariadení. Tvorca musí poskytnúť minimálne jednu komponentu špecifickú pre konkrétne zariadenie alebo produkt [3].

Balíček ovládača musí poskytovať INF (súbor obsahujúci informácie o zariadení) súbor, súbory ovládača a prípadne tiež dodatočné programové komponenty. Zoznam a stručný popis súčastí je uvedený v tab. 2.4.

Tab. 2.4 - Komponenty balíčka ovládača

Súčasť	Popis
súbory ovládača	typicky dynamicky pripájané knižnice s príponou SYS
súbory pre inštaláciu ovládača	súbory s príponou INF obsahujúce informácie používané OS pre inštaláciu podpory zariadenia či produktu
katalóg ovládača	súbor s príponou CAT obsahujúce kryptografický odtlačok každého súboru v balíčku
pomocné inštalátory	dynamicky pripájané súbory Win32 pomáhajúce pri inštalácii
ďalšie súbory	inštaláčny program, ikona zariadenia

Posledným krokom je distribúcia balíčka. Pokiaľ balíček ovládača spĺňa kvalitatívne požiadavky na program Microsoft Windows Logo,²⁷ je ho možné distribuovať cez službu Windows Update [3].

Pomocný inštalátor môže vykonať tieto úlohy:

- prevedenie ďalších úloh pre inštaláciu zariadenia, ktoré nemôže vykonať INF súbor, napr. zápis prídavných informácií, špecifických pre konkrétne zariadenie, do registrov;
- zobrazenie ďalšieho sprievodcu používateľovi po nainštalovaní zariadenia;
- spustenie inštaláčneho programu, ktorý musí bežať v interaktívnom režime po nainštalovaní zariadenia.

2.9.3 Súbory INF pre ovládače

Súbory INF obsahujú informácie a nastavenia zariadenia, napr. ovládač zariadenia alebo program špecifický pre konkrétne zariadenia, ktoré použije systémový inštalátor pre nainštalovanie balíčka ovládača. Možnosti INF súborov sú uvedené a stručne popísané v tab. 2.5.

²⁷ digitálne podpísanie ovládača WHQL (Windows Hardware Quality Labs) podpisom

Tab. 2.5 - Možnosti INF súborov pre inštaláciu zariadení

Súčasť	Popis
ovládač	minimálne jeden ovládač podporujúci zariadenie
inštalátor triedy	spustenie systémového inštalátora pre triedu zariadenia do ktorej zariadenie patrí
pomocný inštalátor	spustenie pomocného inštalátora po nainštalovaní zariadenia
program špecifický pre zariadenie	nainštalovaný inštaláčnym programom, ktorý spustí pomocný inštalátor

Informácie obsiahnuté v INF súboroch:

- názov ovládača a jeho umiestnenie;
- verzia ovládača;
- informácie o záznamoch zapísaných do registrov OS.

Inštaláčné INF súbory neobsahujú inštaláčné skripty. Inštaláčné procedúry vykonáva inštalátor, ako napr. sprievodca novým zariadením (angl. „new device wizard“) alebo sprievodca pridaním hardwaru (angl. „add hardware wizard“).

2.9.4 Podpisovanie ovládača

Podpisovanie balíčka ovládača je možné rozdeliť na prípady:

- **podpísanie balíčka ovládača pre vývoj a testovanie** - používa sa testovací certifikát pre podpísanie verzie balíčka ovládača pred uvedením, pre použitie na testovacích systémoch. Toto umožňuje vývojárom podpísať ovládače vlastnými certifikátmi, nainštalovať ich a vyskúšať ich pri zapnutom overovaní ovládačov v OS Windows.
- **podpísanie vydaného balíčka ovládača pre distribúciu** - po ukončení vývoja a odskúšaní ovládača balíček môže byť podpísaný pre vydanie. Toto identifikuje vydavateľa balíčka ovládača.

Výhody podpisovania vydaných balíčkov ovládačov pre distribúciu:

- zaručuje autentickosť, integritu a spoľahlivosť balíčka, OS Windows používa digitálne podpisy pre overenie identity vydavateľa a overeniu, že balíček ovládača nebol od svojho vydania pozmenený;
- umožňuje najlepšie správanie voči užívateľom vďaka automatickej inštalácii podpísaných ovládačov, ak ovládač nie je podpísaný, inštalátor vyžaduje manuálnu autorizáciu inštalácie ovládača administrátorom systému;²⁸
- umožňuje použitie ovládačov pre režim jadra na 64-bitových verziách OS Windows;²⁹
- prehrávanie určitých druhov digitálneho obsahu³⁰ je povolené, iba ak všetky komponenty OS bežiacie v režime jadra sú digitálne podpísané.

²⁸ tento pridaný krok pri inštalácii má nepriaznivý vplyv aj na prácu pri vývoji a testovaní

²⁹ pre vývoj a testovanie je možné dočasne vypnúť požiadavku na podpísanie ovládača, ale na cieľové systémy používateľov zariadenia alebo produktu nie je vhodné klásť takéto obmedzenia

³⁰ diela chránené DRM – digitálnou správou práv (angl. „digital rights management“)

Uvedené požiadavky platia iba pre OS Windows verzie jadra 6.x, na OS Windows verzie jadra 5.x nie je pre automatickú inštaláciu ovládačov potrebné, aby boli balíčky ovládačov podpísané a taktiež nie je vyžadované podpísanie komponentov bežiacich v režime jadra pre ich zavedenie.

2.10 Zhodnotenie technológií

Z výkonových dôvodov je pre praktické riešenie uprednostňovaná možnosť riešenia v režime jadra, čo ušetrí výmenu dát medzi režimom jadra a režimom používateľa a naspäť. Tým pádom odpadajú riešenia pracujúce v režime používateľa a zostávajú len riešenia založené na tvorbe ovládača pre filtrovanie sieťovej premávky.

Z ekonomických dôvodov nie je možné použiť riešenie NT Kernel WinkFilter. Navyše v tomto prípade by tiež pribudli technické komplikácie pri nasadení.

Pre praktickú realizáciu bolo po zhodnotení vybrané riešenie založené na Windows Filtering Platform. Táto možnosť limituje použitie ovládača na systémy OS Windows verzie jadra 6.x, pretože to je súčasť nového sieťového zásobníka vo Windows verzie jadra 6.x. Využitie na systémoch OS Windows verzie jadra 6.x je ale dostačujúce vzhľadom na plánované nasadenie ovládača len na týchto systémoch.

3 Zvolená metodika riešenia

Vychádzajúc z literárnej rešerše práce a teoretického rozboru riešeného problému bolo pre vlastnú tvorbu ovládača zvolené API Windows Filtering Platform (WFP). Kapitola popisuje použitie tohto rozhrania pre prácu s filtračným strojom OS Windows verzie jadra 6.x a základné logistické prvky nevyhnutné pre ovládač pre OS Windows.

3.1 Štruktúra ovládača WFP

Všeobecná štruktúra ovládača OS Windows je popísaná v kapitole 2.4. Ovládač využívajúci WFP neslúži k práci s fyzickým zariadením a preto musí implementovať iba určité procedúry. Zoznam procedúr je uvedený v tab.3.1, spolu s príznakom využitia WFP ovládačom a krátkym zdôvodnením.

Tab. 3.1 - Zoznam procedúr ovládača a ich využitie WFP ovládačom

Typ procedúry	Využitie ovládačom WFP	Dôvod
Štartovacia procedúra	ÁNO	Potrebná pre zavedenie ovládača do OS, vykonanie inicializácie dátových štruktúr a prepojenia s filtračným strojom OS
Procedúra pre pridanie zariadenia	NIE	Ovládač neobsluhuje žiadne fyzické zariadenie
Odbavovacie procedúry	NIE	Ovládač neobsluhuje žiadne fyzické zariadenie
Procedúra pre začatie I/O operácie	NIE	Ovládač neobsluhuje žiadne fyzické zariadenie
Procedúra pre ošetrovanie prerušenia	NIE	Ovládač neobsluhuje žiadne fyzické zariadenie, spracovanie prerušenia je vykonané sieťovým zásobníkom
Procedúra pre spracovanie prerušenia	NIE	Ovládač potrebuje obdobnú procedúru pre spracovanie dátových jednotiek, tá však nie je volaná správcom I/O, ale filtračným strojom
Procedúry pre dokončenie I/O operácií	NIE	Ovládač potrebuje obdobnú procedúru pre vloženie spracovaných dátových jednotiek naspäť do sieťového zásobníka, tá však nie je volaná správcom I/O, ale filtračným strojom
Procedúry pre zrušenie I/O operácií	NIE	Ovládač by mohol implementovať obdobnú procedúru pre prípad zahodenia toku filtračným strojom

Typ procedúry	Využitie ovládačom WFP	Dôvod
Procedúra pre rýchle odbavenie	NIE	Ovládač neobsluhuje žiadne fyzické zariadenie, všetky I/O operácie sú buď zadávané filtračným strojom, alebo od neho vyžadované
Odstraňovacia procedúra	ÁNO	Potrebná pre uvoľnenie alokovaných zdrojov a zrušenie prepojenia na filtračný stoj OS
Procedúra pre oznámenie vypnutia systému	NIE	Ovládač nevyužíva žiadne prostriedky, ktoré by zabránili úspešnému vypnutiu systému
Procedúry pre záznam chýb	NIE	Možné chyby sú riešené v rámci ovládača, alebo vyplývajú z konfliktu s iným filtrom

WFP ovládač musí ďalej implementovať tzv. procedúry spätného volania (angl. „call-back procedures“), slúžiace pre spracovanie sieťovej premávky zodpovedajúcej zaregistrovaným filtrom a na notifikáciu ovládača o udalostiach, ich zoznam a vysvetlenie použitia je uvedený v tab. 3.2.

Tab. 3.2 - Funkcie spätného volania ovládačov Windows Filtering Platform

Názov prototypu	Úloha
ClassifyFn()	funkcia je volaná operačným systémom keď je objavená sieťová premávka zodpovedajúca niektorému zo zaregistrovaných filtrov
NotifyFn()	funkcia je volaná operačným systémom pre informovanie ovládača o udalostiach asociovaných so spätným volaním, napr. po vložení spracovaných dát naspäť do sieťového zásobníka
FlowDeleteFn()	funkcia je volaná operačným systémom v prípade zrušenia dátového toku v čase jeho spracovania ovládačom; dátové toky sa používajú iba na určitých vrstvách sieťového zásobníka (nad transportnou vrstvou) a táto funkcia nie je povinná pre ovládače pracujúce na vrstvách, ktoré dátové toky nepoužívajú

3.2 Pravidlá pre filtrovanie sieťovej premávky

Ovládač musí byť schopný priradiť vhodné DSCP (kódové pole diferencovaných služieb) značky požadovaným sieťovým tokom. Identifikácia týchto tokov musí byť jednoznačná. Za jednoznačnú identifikáciu sieťového toku je možné považovať súbor nasledujúcich identifikátorov:

- sieťová adresa odosielateľa;
- sieťová adresa prijímateľa;
- protokol transportnej vrstvy;
- port transportnej vrstvy odosielateľa (pri protokoloch využívajúcich porty);
- port transportnej vrstvy prijímateľa (pri protokoloch využívajúcich porty).

Tieto informácie je možné nájsť v literatúre, napr. [29]. Nie všetky identifikátory musia byť špecifikované, v tom prípade sa jedná o všeobecnejšie pravidlo. Samozrejmom súčasťou pravidla je hodnota DSCP, ktorá má byť spracovanému paketu nastavená. Z princípu fungovania filtrovania je hodnota DSCP jediný povinný parameter pravidla.³¹ V tab. 3.3 je uvedený zoznam jednotlivých parametrov pravidiel a ich význam.

Tab. 3.3 - Parametre pravidla pre značkovanie sieťovej premávky

Názov	Typ	Význam
DestinationAddress	REG_STRING	IP adresa prijímateľa
DestinationAddressPrefixLength	REG_DWORD	Dĺžka sieťového prefixu IP adresy cieľa
SourceAddress	REG_STRING	IP adresa odosielateľa
SourceAddressPrefixLength	REG_DWORD	Dĺžka sieťového prefixu IP adresy zdroja
DestinationPort	REG_DWORD	Cieľový port transportnej vrstvy
SourcePort	REG_DWORD	Zdrojový port transportnej vrstvy
Protocol	REG_DWORD	Protokol transportnej vrstvy
DSCP	REG_DWORD	Hodnota DSCP značky

V tab. 3.3 môžu položky IP adres odosielateľa a prijímateľa obsahovať IPv4 alebo IPv6 adresu. Dĺžka sieťového prefixu je 0-32 pre IPv4 a 0-128 pre IPv6, pričom z princípu funkcie hodnoty 0 a maximum pre konkrétnu verziu³² vyjadrujú rovnaké zaobchádzanie – zadaná adresa je adresa konkrétneho zariadenia, nie adresa podsiete.³³ Význam podľa typ je nasledovný:

- **REG_STRING** – reťazec znakov kódovaných Unicode s premenlivou dĺžkou;
- **REG_DWORD** – 32 bitová celočíselná hodnota bez znamienka.

Je potrebné vyriešiť prípad, že konkrétny paket spĺňa podmienky rôznych pravidiel. V takomto prípade sa bežne používajú dve riešenia:

- **Spracovanie podľa najšpecifickejšieho pravidla** – používané napr. pri smerovaní sieťovej premávky;
- **Spracovanie podľa poradia pravidiel**, keď na spracovávanú sieťovú premávku je aplikované prvé pravidlo, ktoré premávka spĺňa – používané napr. v aplikáciách typu firewall.

Pre filtrovanie sieťovej premávky je vhodnejší druhý spôsob – obsluhuje prvým vyhovujúcim pravidlom a preto je pre ovládač použitý. Podobne je táto metóda výrazne jednoduchšia, alternatívna by bola v tomto prípade podstatne zložitejšia. Pri tvorbe pravidiel na to treba pamätať a umiestňovať špecifickejšie pravidlá skôr a všeobecnejšie neskôr.

³¹ v prípade použitia hodnoty DSCP v pravidle ako jediného parametra, je táto DSCP značka nastavená všetkej odchádzajúcej sieťovej premávke

³² t.j. 32 pre IPv4 a 128 pre IPv6

³³ uvažuje sa beztriedne adresovanie

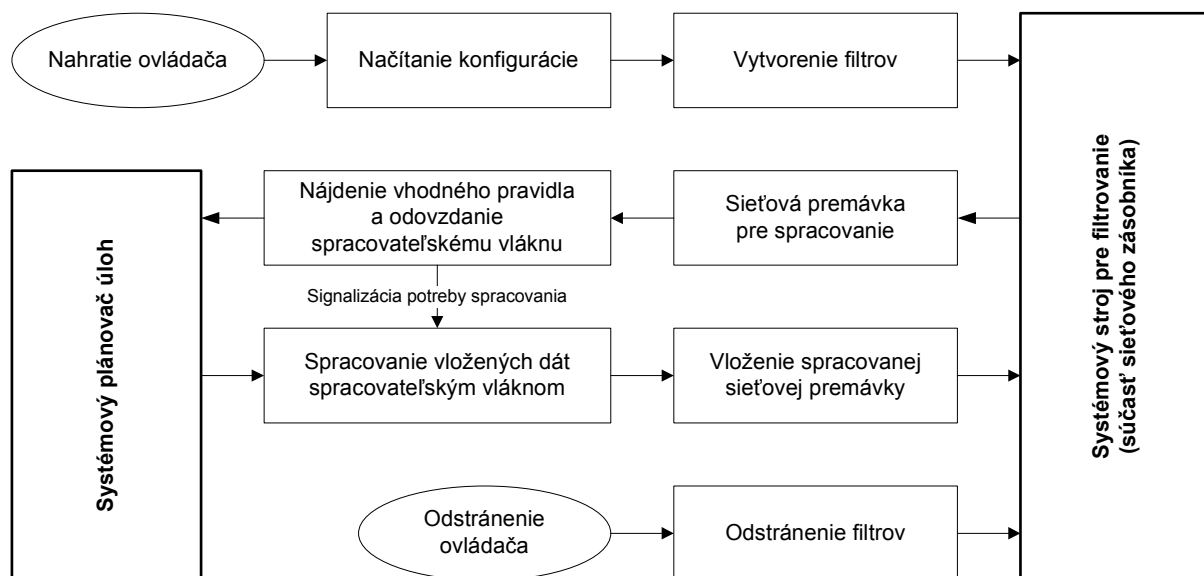
Ďalšou možnosťou filtrovania sieťovej premávky je zaradenie filtrovania na aplikačnej vrstve. Výhodou tohto prístupu je vysoká flexibilita tohto riešenia, veľkou nevýhodou je zložitosť³⁴ a najmä výpočtová náročnosť.

3.3 Návrh štruktúry ovládača pre filtrovanie premávky

Vychádzajúc z kap. 3.1 musí ovládač pre filtrovanie sieťovej premávky s cieľom značkovania DSCP implementovať nasledujúce zo systémom definovaných funkcií:

- **Štartovacia funkcia** (funkcia pre inicializáciu) - DriverEntry();
- **Odstraňovacia funkcia** - DriverUnload();
- **Funkcia volaná pre spracovanie sieťovej premávky** - ClassifyFn();
- **Funkcia volaná pre notifikáciu ovládača o udalostiach** - NotifyFn();

Taktiež môžu byť implementované ďalšie pomocné funkcie, slúžiace na zlepšenie logickej štruktúry a čitateľnosti kódu. Zjednodušená štruktúra ovládača je zobrazená na obr. 3.1.



Obr. 3.1 - Zjednodušená štruktúra ovládača pre filtrovanie sieťovej premávky

3.4 Optimalizácia náročnosti ovládača

Jednou zo základných požiadaviek na ovládač značkujúci sieťovú premávku pre zabezpečenie kvality služby je nízky vplyv samotného ovládača na parametre sieťovej premávky. Preto je otázke výkonu riešenia potrebné venovať náležitú pozornosť.

Pre sieťovú premávku sú z hľadiska zabezpečenia kvality služby dôležité tieto kritériá [29]:

- oneskorenie paketu;
- kolísanie oneskorenia paketov dátového toku;

³⁴ najmä zložitosť implementácie takéhoto riešenia v režime jadra

- stratovosť paketov;
- zmena poradia paketov.

3.4.1 Vplyv ovládača na oneskorenie paketu a kolísanie oneskorenia

Ovládač pre značkovanie sieťovej premávky z princípu svojho fungovania vnáša oneskorenie do prenosového reťazca. Každý paket je nutné zachytiť, nájsť prislúchajúcu DSCP značku, vykonať zmeny a následne znovu vložiť do sieťového zásobníka, čo zaberie určitý čas. Dôležitá je minimalizácia tohto času. Určité možnosti zmeny existujú, sú popísané v kap. 3.4.4.

Vplyv ovládača na oneskorenie paketu je všeobecne malý, konkrétne informácie je možné nájsť v kap. 3.4.7. Doba spracovania každého paketu ale môže rásť pri vysokom zaťažení systému. Podobne pri kolísaní zaťaženia môže dochádzať k rozdielom v dobách spracovania rôznych paketov patriacich tomu istému toku. Keďže OS Windows nie je systémom reálneho času (angl. „real-time system“),³⁵ v prípade systémových programov sa používa aj názov operačný systém reálneho času (angl. „real-time operating system“, RTOS – operačný systém reálneho času), technicky nie je možné zaručiť maximálnu dobu spracovania paketu a stálosť tejto doby. Pri bežnom zaťažení systému je ale vplyv na oneskorenie paketu malý a stabilita tohto vloženého oneskorenia medzi rôznymi spracovaniami približne konštantná.

Volané systémové funkcie, konkrétne `FwpsConstructIpHeaderForTransportPacket0()`, môžu vykazovať vyššiu dobu spracovania v prípade, že protokolom transportnej vrstvy je TCP (protokol s riadením vysielania) [14]. Reálny dopad ale nie je veľký, nakoľko protokolom TCP sa väčšinou prenášajú dáta vo väčších zhlukoch³⁶ a tak volanie tejto funkcie je málo časté. Iné protokoly, ktoré vysielajú dáta častejšie zase netrpia podobným spomalením.

3.4.2 Vplyv ovládača na stratovosť paketov a zmenu ich poradia

Ovládač kvôli svojej architektúre nemení poradie spracovávaných paketov v dátovom toku, pakety určené pre značkovanie sú radené do FIFO (prvý dnu prvý von) fronty a teda sú spracovávané v poradí, v akom boli dodané na spracovanie ovládaču sieťovým zásobníkom.

Kvôli minimalizácii vplyvu dĺžky spracovania paketov na celú sieťovú komunikáciu sú ale pakety určené na spracovanie značkované a vkladané v druhom vlákne a klasifikačná funkcia odovzdáva riadenie kódu sieťového zásobníka v krátkom čase. Z toho dôvodu sú značkované pakety odoslané neskôr ako iné pakety, ktoré nespĺňali žiadne z pravidiel.

Stratovosť paketov je ovládačom zvyšovaná iba v extrémnych situáciách. Z princípu funkcie, keď značkovaná sieťová premávka je najprv vyradená z bežného toku spracovania, upravená a znovu vložená do sieťového zásobníka vyplýva, že pokiaľ dôjde k zlyhaniu v časti,

³⁵ systém v ktorom je pre každú úlohu určená časová hranica, do ktorej musí byť úloha skončená a teda zabezpečujúci spracovanie úloh v známom maximálnom čase

³⁶ výnimkou sú protokoly poskytujúce vzdialený terminál, ako napr. TELNET alebo SSH, v tomto prípade sú ale jednotlivé pakety od seba časovo dostatočne vzdialené

keď už bol paket z toku odstránený a predtým ako bol znovu vložený po úprave, tak príde k strate paketu. Takáto situácia môže vzniknúť napr. pri kritickom nedostatku pamäťových prostriedkov, pri bežnej prevádzke je málo pravdepodobná.

Dôležitou vlastnosťou je aj vplyv ovládača na sieťovú premávku ako celok, teda na sieťovú premávku, ktorá ovládačom spracovaná nie je. Vďaka architektonickým rozhodnutiam je vplyv ovládača na premávku, ktorá nespĺňa žiadne pravidlo v podstate nulový.

3.4.3 Ovplyviteľné sekcie ovládača

Ovládač veľký objem svojho času volá služby operačného systému a výkon v týchto prípadoch nie je možné ovplyvniť zmenou návrhu ovládača. Možnosti pre zmenu návrhu sú vo vyhľadávaní pravidla, ktoré aktivovalo spracovanie aktuálneho dátového toku, teda pre každý spracovávaný paket. Vyhľadávaciemu algoritmu a možnostiam jeho optimalizácie sa podrobnejšie venuje kap. 3.4.4. Podobne možno robiť zmeny v architektúre spracovateľského vlákna, napríklad ho prispôbiť viacprocesorovým systémom. Architektúre spracovateľského vlákna a jej optimalizácii je venovaná kap. 3.4.5.

V ohľade pamäťovej náročnosti je možné robiť zmeny najmä v množstve pamäte vyhradenej pre pravidlá. Tieto prostriedky sú dynamicky alokované v nestránkovateľnej pamäti, ktorej množstvo je obmedzené a jedná sa o cenný systémový prostriedok [7], a uvoľnené sú až pri ukončení behu ovládača. Počas behu sú dočasne alokované ďalšie pamäťové bloky v nestránkovateľnej pamäti, často väčšie, ale na kratšiu dobu a obsahujú len nutne potrebné informácie. Pamäťovej náročnosti ovládača a jej optimalizácii sa bližšie venuje kap. 3.4.6.

Ďalšou možnosťou optimalizácie je kontrola pravidiel pri inicializácii. Tu treba vyriešiť problém protichodných požiadaviek, robustnosti a výpočtového výkonu. Čím hlbšia kontrola pravidiel je vykonaná, tým pomalšia je inicializácia a teda aj čas potrebný pre reštartovanie ovládača po zmene pravidiel. Preto je vhodné presunúť väčšiu časť kontroly do API pre manipuláciu pravidiel a priamo v ovládači vykonávať len základnú kontrolu. Súvisiacim problémom je výber načítavania pravidiel len z registrov OS, čo znamená reštart ovládača po každej zmene pravidiel a vytvára krátke okno, kedy sieťová premávka nie je značkováná, alebo operatívnu zmenu s využitím medziprocesnej komunikácie, čo znamená zmenu pravidiel bez reštartu. Druhá možnosť sa javí ako lepšie riešenie, avšak tento prípad vyžaduje výrazné zvýšenie zložitosti spracovateľskej logiky a pamäťových nárokov. Môže totiž nastať prípad, že paket už zaradený na spracovanie nezodpovedá žiadnemu existujúcemu pravidlu.³⁷ Taktiež je nutné riešiť zmenu identifikátorov filtrov aj pre rovnaké pravidlá. Posledným problémom je to, že počas toku by nemalo dôjsť k zmene triedy zabezpečenia kvality služby, nakoľko výsledné správanie sieťovej infraštruktúry k takémuto toku je ťažko možné predvídať.

³⁷ pravidlo, podľa ktorého bol na spracovanie zaradený bolo z novej sady odstránené

3.4.4 Vyhľadávací algoritmus

Vyhľadávací algoritmus je spustený pre každý spracovávaný paket, aby bolo nájdené pravidlo, ktorému aktuálne spracovávaný paket zodpovedá. Výkon vyhľadávacieho algoritmu je ovplyvnený počtom pravidiel, ktoré ovládač zaregistroval. Keďže sa v praktických podmienkach predpokladá relatívne málo pravidiel, približne jedna až dve desiatky, bol ako vyhľadávací algoritmus zvolený algoritmus s lineárnou časovou zložitou v notácii „O“ [18], prehľadávajúci jednotlivé pravidlá v poradí. Kód v jazyku C vyzerá nasledovne:

```
// Find rule by filterID
for (i = 0; i < ruleCount; ++i) {
    if (QoSRuleList[i].filterID == packet->filterID) {
        rule = &QoSRuleList[i];
        IPV6 = FALSE;
        break;                // we have a winner
    }
    if (QoSRuleList[i].filterIDv6 == packet->filterID) {
        rule = &QoSRuleList[i];
        IPV6 = TRUE;
        break;                // we have a winner
    }
}
ASSERT(rule != NULL);
```

V prípade, že reálne použitie bude využívať výrazne vyšší počet pravidiel, je možné zameniť tento algoritmus za napr. algoritmus založený na binárnom strome, ktorý má logaritmickú zložitou v notácii „O“ [18]. Nevýhodou tohto riešenia by ale boli vyššie pamäťové nároky, menší výkon v prípade malého počtu pravidiel, dlhšia doba inicializácie ovládača a asi najväčším problémom je potreba kompletnej implementácie nového dátového typu (binárneho vyhľadávacieho stromu) v režime jadra.

3.4.5 Spracovateľské vlákno

Spracovateľské vlákno je použité z dôvodu zrýchlenia návratu z funkcie ovládača volanej sieťovým zásobníkom a tým pádom zmenšenia vplyvu ovládača na sieťový zásobník OS ako celok a jeho výkon. Priamo v obslužnej funkcii sa preto vykonáva iba nutné minimum spracovania.

Tento systém je podobný odloženému spracovaniu procedúr (angl. „deferred procedure call“, DPC), čo je systém využívaný napríklad pri obsluhu prerušení umožňujúci odložiť nízko prioritné časti obsluhy prerušenia na neskôr a tým zvýšenie odozvy systému. Rozdiel je v tom, že logika používaná v rámci ovládača je výrazne jednoduchšia a je obsluhovaná ovládačom samotným, nie operačným systémom.

Realizácia spracovateľského vlákna je v podstate možná niekoľkými variantmi:

- Spracovateľské vlákno je spúšťané obslužnou rutinou pri prijatí sieťovej premávky a samo sa ukončí po vyprázdnení fronty. Tento spôsob je jednoduchý, ale pridáva režijné náklady vôli spúšťaniu spracovateľského vlákna z obslužnej rutiny.
- Spracovateľské vlákno spustené pri inicializácii ovládača vykonáva dáta zo zoznamu úloh, kým ho nevyprázdni, kedy sa následne uspí a ďalej nie je spúšťané. Obslužná rutina

spracovania premávky preto kontroluje stav spracovateľskej fronty. Ak sa v nej nachádza minimálne jedna položka, tak ďalšiu úlohu len vloží na koniec. Ak je zoznam prázdny, tak po vložení úlohy zašle signál potreby spracovania spracovateľskému vláknu. Tento spôsob je výhodný, pretože je jednoduchý, výpočtovo a pamäťovo nenáročný a poskytuje dobrý výkon. Problém nastáva na viacprocesorových systémoch v dvoch prípadoch. Prvá možnosť je vysoké sieťové zaťaženie systému ako celku a vysokom množstve procesorov,³⁸ čo by spôsobilo vyšší prísun nových úloh, ako je vlákno bežiacie na jednom procesore schopné priebežne spracovávať. Druhým problémom je to, že ak obsluháva rutinu spracovania a spracovateľské vlákno nebežia na jednom procesore viacprocesorového systému, tak dôjde ku zbytočnému kopírovaniu dát, čo má nepriaznivý vplyv na výkon.

- Pre minimalizáciu vplyvu kopírovania by bolo možné použiť prvú variantu s pridaním plánovania spracovateľského vlákna pre beh na konkrétnom procesore, ale to by viedlo k systému, ktorý nevie zaručiť zachovanie poradia paketov a vnáša kolísanie oneskorenia jednotlivých paketov.
- Z hľadiska škálovania je optimálnym riešením vytvorenie niekoľkých spracovateľských vlákien v čase inicializácie ovládača, pre každý procesor jedno a naplánovaných tak, aby bežalo len jedno na konkrétnom procesore. Každé vlákno by spracovávalo požiadavky, ktoré boli sieťovým zásobníkom predané ovládaču na spracovanie na konkrétnom procesore. Táto možnosť ale znižuje výkon na jednoprocessorových systémoch kvôli zložitejšej logike a jej výhoda sa prejavuje až na systémoch s relatívne vysokým počtom procesorov. Zároveň má zložitú signalizáciu a je náchylná na zámenu poradia paketov, prípadne zvýšenie rozptylu kolísania oneskorenia.³⁹ Jedná sa ale o pravé paralelné spracovanie.

3.4.6 Pamäťová zložitosť

Vzhľadom k tomu, že ovládač musí svoje požiadavky na pamäťové prostriedky uspokojovať z nestránkovanej pamäte, trvalo prístupnej v operačnej pamäti [7], je vhodné minimalizovať spotrebu pamäte ovládačom.

V prípade predávania dát medzi spracovateľskou rutinou a spracovateľským vláknom je preto vhodné ukladať len potrebné dáta a v prípade ukladania pravidiel, ktoré sú v pamäti prítomné počas celej doby behu ovládača, je podobne vhodné ukladať iba nutne potrebné informácie.

Pri inicializácii ovládača je vo fáze pridávania filtračných pravidiel do sieťového zásobníka potrebné uchovávať kompletnú logickú štruktúru pravidiel. Následne sú ale tieto informácie zbytočné, nakoľko spracovateľská rutina dostáva identifikátor pravidla a nie

³⁸ v OS Windows verzie jadra 6.x je možné spracovávať sieťovú premávku na ľubovoľnom počte procesorov na rozdiel od jediného v predchádzajúcich verziách

³⁹ v prípade nerovnakej záťaže jednotlivých procesorov

priamo parametre. Po zaregistrovaní pravidiel stačí ukladať identifikátor pravidla (8 oktetov) a DSCP značku (6b = 1 oktet), t.j. dohromady 9 oktetov na jedno pravidlo.

Ďalšia možnosť je ponechanie kompletných pravidiel v pamäti, čo spôsobuje mierne nižšiu réžiu pri inicializácii, ale aj vyššiu pamäťovú náročnosť počas celého behu ovládača.

3.4.7 Meranie vplyvu ovládača na parametre sieťovej premávky

Reálne meranie vplyvu je dôležitou súčasťou hodnotenia výkonu ovládača pracujúceho so sieťovou premávkou. Pre meranie je možné využiť bežné nástroje pre syntetické testovanie výkonu siete, ale dôležité je aj zhodnotenie vplyvu na reálne aplikácie. Pre meranie výkonu by sa mal používať ovládač s voľným zostavením, pretože kontrolované zostavenie degraduje výkon kódu.

Pre meranie výkonu bola vytvorená jednoduchá sada skriptov postavených nad nástrojmi IPERF, popísaný v [27] a D-ITG, popísaný v [28]. Meranie je vytvorené tak, že sú vyžadované dve stanice, jedna s nich je server spracovávajúci výsledky, druhá klient, na ktorom testovaný ovládač beží. Vykonávanie skriptov je automatizované, výsledky sú zapísané do dátového súboru. Spracovanie výsledkov je potrebné vykonať ručne.

Boli vykonané iba jednoduché merania potvrdzujúce, že ovládač v súčasnej konfigurácii má len malý negatívny vplyv na parametre sieťovej premávky. Komplexné merania neboli realizované pre nedostatok informácií o plánovanom nasadení riešenia. Je ale vhodné ich vykonať pri plánovaní nasadenia, nakoľko potom už budú k dispozícii presnejšie požiadavky.

3.5 Aplikačné rozhranie pre riadenie ovládača

Programové rozhranie pre riadenie ovládača je dôležitou súčasťou riešenia. Poskytuje jednoduchý spôsob správy pravidiel a ovládanie stavu ovládača. Základné požiadavky na API ovládača sú nasledovné, podrobnejšie informácie je možné získať v [21]:

- poskytovanie funkcionality ovládača logickým spôsobom s prihliadnutím na použitie – rozhranie by malo byť jednoduché a skrývať zložité detaily práce s ovládačom, poskytované funkcie by mali jednoduchým spôsobom umožňovať vykonávať bežné operácie nad ovládačom;
- flexibilita rozhrania a široká škála použiteľných jazykov pre tvorbu samotnej aplikácie – API by nemalo limitovať aplikačných programátorov a malo by umožňovať použitie v čo najširšej škále programovacích jazykov a prostredí;
- minimalizácia zamykania a potreby synchronizácie, obmedzenie len na nutné prípady;
- poskytovanie výkonu vhodného pre aplikácie aj pre ovládač – optimalizácia výkonu rozhrania by mala zodpovedať jeho dôležitosť pri práci a reálnemu dopadu na výkon systému⁴⁰ ako celku;

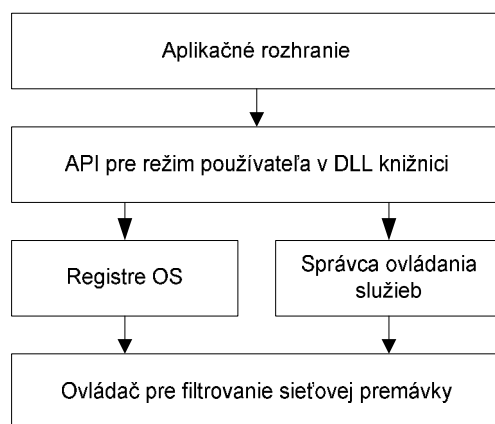
⁴⁰ v tomto prípade sa systém myslí vytváraný systém, čiže ovládač + aplikačné rozhranie + aplikácia využívajúca služby ovládača

- rozhranie nesmie za žiadnych okolností viesť k pádu aplikácie či systému alebo narušeniu bezpečnosti, ani v prípade nevhodného použitia.

Tieto požiadavky prinášajú zaujímavé implikácie. Asi najzjavnejším miestom, kde sa prejavujú, je voľba aplikačného rozhrania a vonkajšia forma API. Aj keď teoreticky existuje niekoľko možných riešení,⁴¹ v praxi je pre splnenie požiadavku na flexibilitu použiteľná len forma DLL knižnice s funkciami s čistým C rozhraním [21].

3.5.1 Technické prostriedky aplikačného rozhrania

Aplikačné rozhranie je využívané aplikáciami, ale k svojej činnosti tiež používa ďalšie aplikačné rozhrania poskytované operačným systémom. Jedná sa o systémové funkcie pre prácu s hodnotami v registroch a pre API pre prácu so správcom ovládania služieb. V registroch sú ukladané pravidlá pre ovládač, služby správcu ovládania služieb sú využívané na spúšťanie, zastavovanie a reštartovanie ovládača. Na obr. 3.2 sú znázornené vrstvy aplikačného rozhrania.



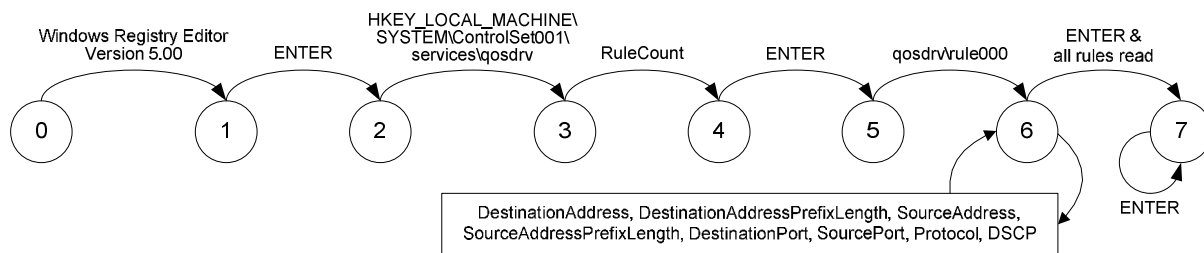
Obr. 3.2 - Vrstvy aplikačného rozhrania pre riadenie ovládača

Aplikačné rozhranie má vonkajšiu štruktúru v jazyku C, ale implementácia využíva prvky jazyka C++, podrobne popísané v [17] a dátové kontajnery jeho štandardnej knižnice, popísané v [18].

Pre uloženie pravidiel v pamäti je vhodný dátový kontajner vektor, čo je v podstate dynamicky rastúce pole. Popis kontajnera vektor a jeho vlastností a použitia je možné nájsť v [18], praktické ukážky použitia v [17]. V podstate sa jedná o výrazné zjednodušenie práce kvôli odstráneniu réžie dátového úložiska.

Pre načítanie REG súborov s pravidlami sú využívané funkcie štandardnej knižnice jazyka C++ pre spracovanie reťazcov, podrobne popísané v [18]. Práca s nimi je výrazne jednoduchšia ako v prípade funkcií pre spracovanie reťazcov štandardnej knižnice jazyka C, umožnili vytvorenie kratšieho a lepšie čitateľného kódu, ktorý má aj menšie pamäťové nároky. Stavový automat pre načítanie REG súboru je zobrazený na obr. 3.3.

⁴¹ napr. .NET rozhranie, COM / ActiveX rozhranie



Obr. 3.3 - Stavový automat načítania REG súboru

Každé pravidlo je kontrolované na logickú súdržnosť. Pre kontrolu sieťových adries boli využité funkcie Winsock, ostatné časti sú kontrolované vytvoreným vlastným kódom.

3.5.2 Štruktúra hlavičkového súboru API

Vzhľadom na definované požiadavky vychádza nasledujúca štruktúra hlavičkového súboru:

```

#ifdef __cplusplus
extern "C" {
#endif
#ifdef DLL_BUILD
#pragma push_macro("DCC")
#undef DCC
#endif
#ifdef _WIN64
#define DCC __stdcall
#else
#define DCC
#endif
#pragma push_macro("IMPEXP")
#undef IMPEXP
#ifdef DLL_BUILD
#define IMPEXP __declspec(dllexport)
#else
#define IMPEXP __declspec(dllimport)
#endif
IMPEXP void APIENTRY DCC functionA();
IMPEXP void APIENTRY DCC functionB();
IMPEXP void APIENTRY DCC functionC();
#ifdef __cplusplus
}
#endif
#pragma pop_macro("IMPEXP")
#ifdef DLL_BUILD
#pragma pop_macro("DCC")
#endif

```

Podrobnejšie informácie o konkrétnych dôvodoch tejto štruktúry je možné nájsť v [22] a [23]. Výhoda tejto štruktúry je taká, že umožňuje jeden hlavičkový súbor využiť pre finálne zostavenie aj pre zostavenie pre ladenie, ako deklaráciu funkcií pri tvorbe knižnice, ale aj pri tvorbe aplikácie, a tiež rieši rozdiely medzi 32-bitovými a 64-bitovými systémami. Rozhranie je v jazyku C kvôli portabilite a rozšíreniu možností použitia, ale pre implementáciu knižnice je možné využiť aj jazyk C aj C++.

4 Sieťový ovládač pre značkovanie premávky

Praktický výstup práce je možné rozdeliť do niekoľkých častí:

- vlastný ovládač pre značkovanie sieťovej premávky;
- podporné súbory pre ovládač;
- API pre režim používateľa slúžiace k nastavovaniu parametrov a riadeniu ovládača;
- aplikácia pre riadenie ovládača využívajúca toto API;
- systém pre meranie výkonu ovládačov filtrujúcich sieťovú premávku.

4.1 Použitie vytvoreného ovládača

Pre inštaláciu a odinštalovanie ovládača je možné použiť vytvorené skripty. Následné spustenie a zastavenie je možné vykonávať s pomocou vytvoreného API, popis vhodných funkcií je v kap. 4.3.4, alebo z príkazového riadku napr. pomocou príkazu „net“ nasledovne:

- spustenie ovládača:

```
net start qosdrv
```

- zastavenie ovládača:

```
net stop qosdrv
```

Či je ovládač nainštalovaný, je možné zistiť pomocou dodaného aplikačného rozhrania, z príkazového riadku len nepriamo, ak príkaz pre spustenie ovládača zlyhá s chybou, že zadaná služba neexistuje. Podobným spôsobom je možné z príkazového riadku zistiť, či ovládač beží alebo nie.

4.2 Ovládač pre značkovanie paketov

Jedná sa o implementáciu štruktúry navrhutej v kap. 3.1 a kap. 3.3. Okrem súborov technického zabezpečenia⁴² je táto časť rozdelená do troch súborov:

- QoS_drv.c – obsahuje implementáciu podpornej logiky;
- QoS_marking.h – obsahuje spoločné deklarácie;
- QoS_marking.c – obsahuje vlastnú značkovaciu logiku.

4.2.1 Vstupná a výstupná funkcia ovládača

Pozostáva z funkcií umožňujúcich inicializáciu ovládača po jeho nahratí do pamäte a prípadné operácie potrebné pred zastavením a odstránením ovládača z pamäte. Taktiež ale načítanie parametrov z registrov a ich validáciu, či vytvorenie prepojení na sieťový zásobník pre extrakciu podľa filtračných podmienok a vkladanie upravených paketov.

⁴² projektový súbor, súbory pre zostavenie

Prvou časťou tvorby ovládača je deklarácia vstupnej a výstupnej funkcie ovládača. Ich názvy sú voliteľné,⁴³ predpis funkcií je však určený. Definícia vyzerá nasledovne:

```
// DriverEntry & DriverUnload
DRIVER_INITIALIZE DriverEntry;
NTSTATUS DriverEntry(IN PDRIVER_OBJECT driverObject, IN PUNICODE_STRING
    registryPath);

DRIVER_UNLOAD DriverUnload;
VOID DriverUnload(IN PDRIVER_OBJECT driverObject);
```

DriverEntry()

Nasleduje popis štruktúry vstupnej funkcie ovládača. Funkcia má dva parametre:

- IN PDRIVER_OBJECT driverObject – ukazovateľ na štruktúru DRIVER_OBJECT, ktorej inštancia v OS reprezentuje ovládač, musí byť ovládačom inicializovaná ukazovateľmi na funkcie vykonávajúce jednotlivé časti;
- IN PUNICODE_STRING registryPath – ukazovateľ na Unicode reťazec znakov obsahujúci cestu ku kľúču (angl. „registry key“) ovládača v registroch.

Návratová hodnota je typu NTSTATUS,⁴⁴ hodnota STATUS_SUCCESS oznamuje OS, že inicializácia ovládača prebehla v poriadku, iná hodnota znamená chybu a odstránenie ovládača z pamäte. V takomto prípade sa predpokladá, že ako reakcia na chybu prebehlo uvoľnenie alokovaných prostriedkov. Štruktúra funkcie pre inicializáciu je zobrazená na obr. 4.1.

Ako prvá časť je vykonané nahratie parametrov z registrov a ich validácie. K tomuto účelu je použitá podporná funkcia QoSLoadConfig(), jej popis nasleduje neskôr. Ďalším krokom je vytvorenie zariadenia v systéme, ktoré reprezentuje ovládač. K tomuto je využité volanie OS IoCreateDevice(). Nasleduje vytvorenie vstupných bodov pre vkladanie modifikovaných paketov naspäť do sieťového zásobníka. K tomuto účelu sa využívajú dve volania systémovej funkcie FwpsInjectionHandleCreate0(), jedno pre IPv4 a jedno pre IPv6. Vstupné body sa nachádzajú na sieťovej vrstve pre obe varianty.

Ďalej je vytvorený a inicializovaný systémom spravovaný zoznam úloh pre spracovateľské vlákno.⁴⁵ K tomuto účelu slúži nasledujúci kód:

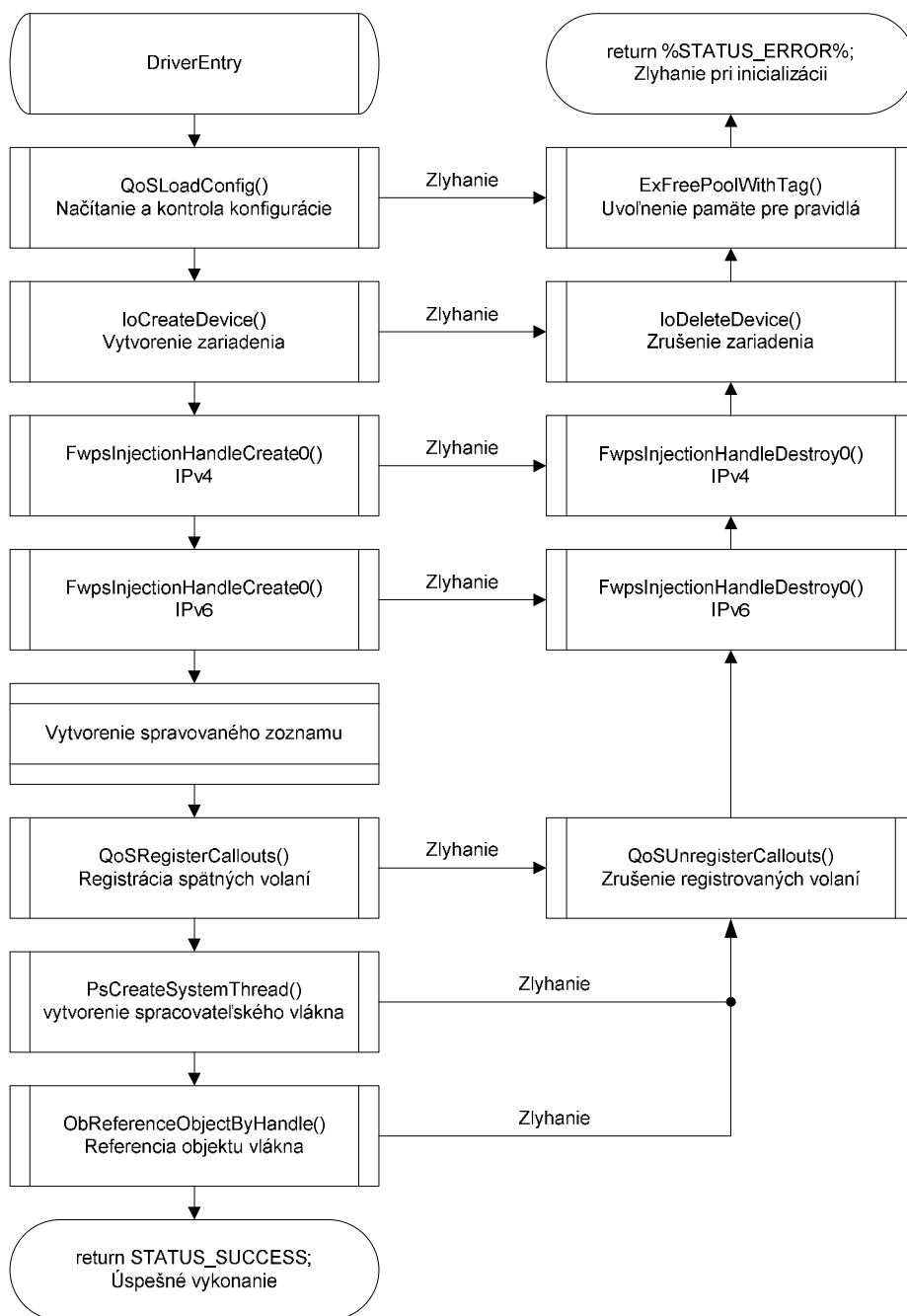
```
InitializeListHead(&gPacketQueue);
KeInitializeSpinLock(&gPacketQueueLock);
KeInitializeEvent(&gPacketQueueEvent, NotificationEvent, FALSE);
```

Ako ďalšia nasleduje registrácia spätných volaní (angl. „callout“), ktorú vykonáva podporná funkcia QoSRegisterCallouts(). Ďalej dochádza k vytvoreniu druhého vlákna vykonávajúceho kód funkcie QoSWorker(), slúžiacej pre spracovanie odchytenej sieťovej premávky. Ukazovateľu na objekt druhého vlákna je po jeho vytvorení zvýšený počet referencií, aby objekt nebol automaticky zmazaný operačným systémom.

⁴³ používajú sa ale názvy, ktoré sú nepísaným štandardom, DriverEntry pre vstupnú funkciu a DriverUnload pre výstupnú funkciu

⁴⁴ v podstate 32bitová hodnota s definovaným významom jednotlivých kombinácií

⁴⁵ obsiahnuté vo funkcii QoSWorker()

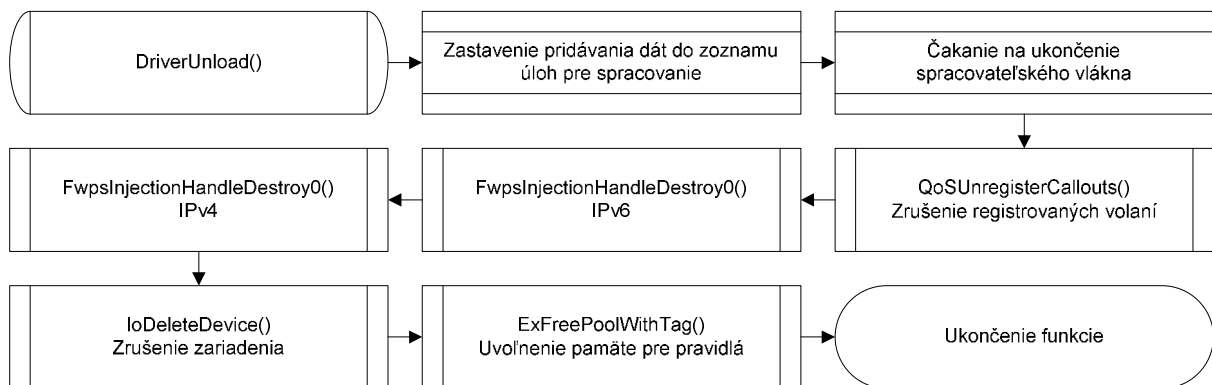


Obr. 4.1 - Štruktúra funkcie pre inicializáciu ovládača

Záverečná časť funkcie pre inicializáciu sa vykonáva iba v prípade výskytu chyby. Služí na uvedenie systému do pôvodného stavu. Rozsah návratových operácií (angl. „roll-back“) závisí na fáze, v ktorej chyba nastala. Môžu byť zrušená registrácia spätných volaní pomocnou funkciou QoSUnregisterCallouts(). Ak chyba nastala po vytvorení vstupných bodov pre spätné vkladanie modifikovanej sieťovej premávky, tak sú tieto tiež zrušené dvomi volaniami systémovej funkcie FwpsInjectionHandleDestroy0(). Tiež môže byť vykonané zrušenie vytvoreného systémového zariadenia systémovou funkciou IoDeleteDevice() a nakoniec zrušenie dynamicky alokovanej pamäte pre načítané pravidlá s pomocou systémovej funkcie ExFreePoolWithTag().

DriverUnload()

Funkcia pre odstránenie ovládača je volaná OS v prípade pokynu k ukončeniu behu ovládača, jej priebeh je znázornený na obr. 4.2. Funkcia má jeden parameter, ktorým je ukazovateľ na objekt reprezentujúci ovládač a nevracia žiadnu hodnotu. Za úlohu má uvoľniť prostriedky alokované ovládačom pri jeho inicializácii a počas behu. Na začiatku zastaví pridávanie úloh do zoznamu pre spracovateľské vlákno a toto vlákno následne po spracovaní už zaradených úloh ukončí. Následne sa volá pomocná funkcia QoSUnregisterCallouts() slúžiaca na odstránenie pravidiel zo systémového stroja pre filtrovanie. Ďalej sú uvoľnené prostriedky alokované funkciou pre inicializáciu – zrušenie vstupných bodov pre upravenú sieťovú premávku, zmazanie objektu zariadenia a uvoľnenie pamäte pre pravidlá filtrovania. Táto časť je podobná správaniu funkcie pre inicializáciu v prípade chyby.



Obr. 4.2 - Štruktúra funkcie pre odstránenie ovládača

4.2.2 Podporná logika ovládača

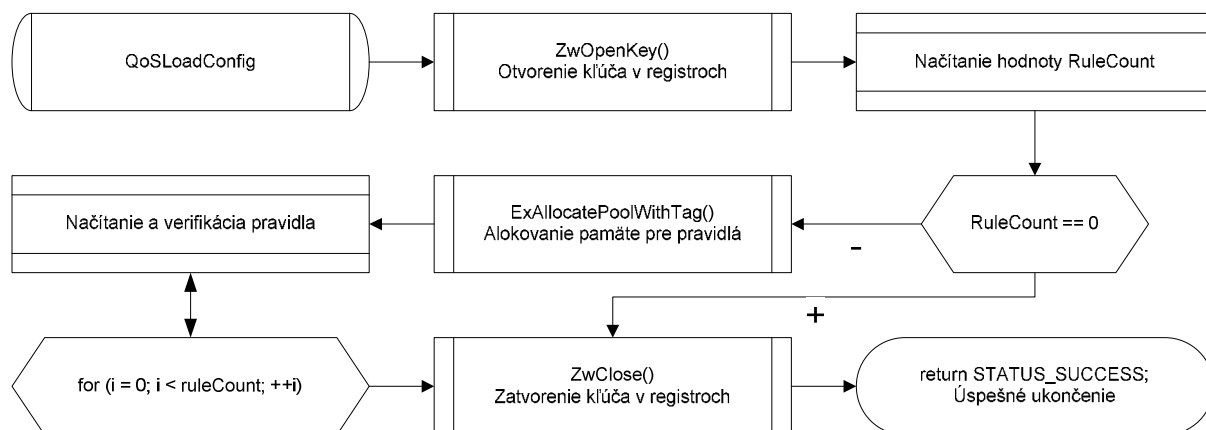
QoSLoadConfig()

Prvou volanou pomocnou funkciou z funkcie pre inicializáciu je QoSLoadConfig(). Táto sa stará o načítanie pravidiel z registrov OS a základnú kontrolu ich rozumnosti. Predpis funkcie je nasledovný:

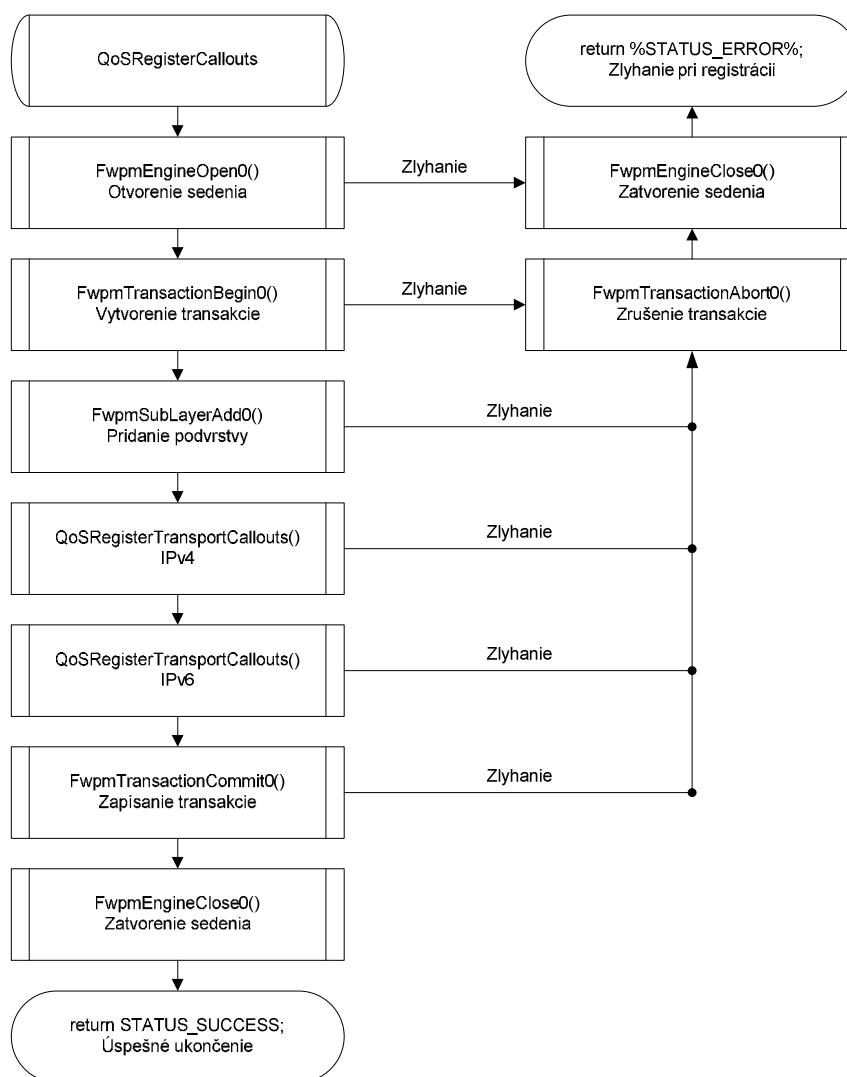
```
NTSTATUS QoSLoadConfig(IN PUNICODE_STRING registryPath);
```

Parameter funkcie je ukazovateľ na reťazec obsahujúci cestu ku kľúču s nastaveniami ovládača v registroch OS. Funkcia z registrov zistí počet pravidiel a následne pre ne pomocou systémového volania ExAllocatePoolWithTag() dynamicky alokuje pamäť v nestránkovanom zásobníku.⁴⁶ Ďalej sú v slučke načítané parametre jednotlivých pravidiel pre filtrovanie. Pravidlá pre filtrovanie sú popísané v kap. 3.2. Štruktúra funkcie je na obr. 4.3.

⁴⁶ takto alokovaný priestor je trvalo prístupný v operačnej pamäti počítača a je teda prístupný bez ohľadu na aktuálny IRQL stupeň procesora vykonávajúceho kód ovládača



Obr. 4.3 - Štruktúra funkcie pre načítanie pravidiel



Obr. 4.4 - Štruktúra funkcie pre registráciu popisov

QoSRegisterCallouts()

Ďalšou pomocnou funkciou je QoSRegisterCallouts(), slúžiaca na registráciu popisov v stroji pre filtrovanie. Funkcia má jeden parameter a to ukazovateľ na objekt zariadenia. Registrácia filtrov do sieťového zásobníka je vykonaná ako transakcia, takže ak dôjde ku chybe, stav

filtračného stroja je rovnaký ako pred začiatkom zmien, celá transakcia sa vykoná až keď všetky čiastkové úlohy prebehnú v poriadku. Predpis funkcie je nasledovný:

```
NTSTATUS QoSRegisterCallouts(IN void* deviceObject);
```

Na začiatku behu je otvorené sedenie (angl. „session“) do systémového filtračného stroja funkciou OS FwpmEngineOpen0(), následne je začatá nová transakcia vo filtračnom stroji funkciou FwpmTransactionBegin0(). Od tej chvíle je možné sa zrušením transakcie vrátiť presne do stavu pred začatím transakcie. Ako prvá udalosť v rámci transakcie je pridanie novej podvrstvy vo filtračnom stroji funkciou FwpmSubLayerAdd0(). Následne sú zaregistrované popisné ovládače (angl. „callout drivers“) pre transportnú vrstvu. Pre túto úlohu je volaná pomocná funkcia QoSRegisterTransportCallouts(). Na záver je transakcia uzavretá funkciou FwpmTransactionCommit0() a všetky zmeny sú zapísané. Štruktúra funkcie QoSRegisterCallouts() je zobrazená na obr. 4.4.

Zobrazená štruktúra zodpovedá plne funkčnej verzii. Kvôli chybe v IPv6 vetve je táto vetva vypnutá a funkcia QoSRegisterTransportCallouts() nie je pre IPv6 volaná. To spôsobí efektívne ignorovanie IPv6 adries. Po vyriešení problémov s IPv6 vetvou stačí znovu uviesť do prevádzky časť volajúcu QoSRegisterTransportCallouts() pre IPv6.

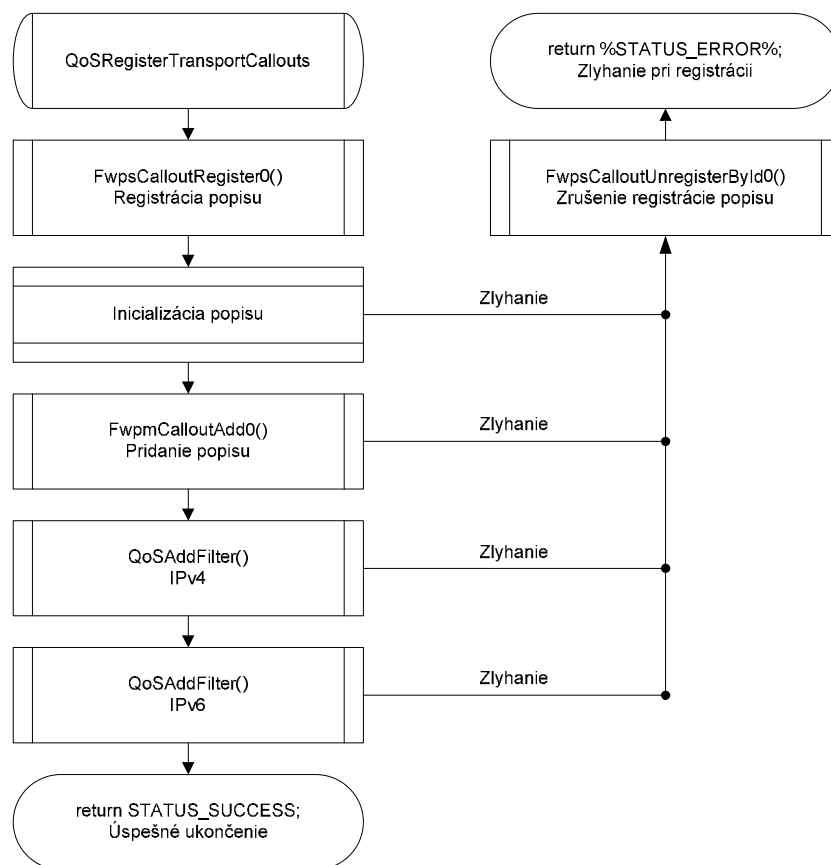
QoSRegisterTransportCallouts()

Pomocnou funkciou slúžiacou na registráciu popisných ovládačov pre transportnú vrstvu je QoSRegisterTransportCallouts(). Predpis funkcie je nasledný:

```
NTSTATUS QoSRegisterTransportCallouts(IN const GUID* layerKey, IN const GUID  
*calloutKey, IN void *deviceObject, OUT UINT32 *calloutId);
```

Na začiatku je zaregistrovaný popis (angl. „callout“) funkciou FwpsCalloutRegister0(), následne je inicializovaná štruktúra popisu a popis je pridany do systému funkciou FwpmCalloutAdd0(). Ďalej sú v cykle pridané filtre využitím pomocnej funkcie QoSAddFilter(). V prípade, že pri vykonávaní funkcie dôjde k chybe, je registrácia popisu zrušená volaním funkcie FwpsCalloutUnregisterById0().

Parametrami funkcie sú ukazovateľ na globálne unikátny identifikátor (angl. „globally unique identifier“), GUID, pridanej podvrstvy, potrebný pre pridávanie filtrov, GUID popisu, pod ktorým sa má zaregistrovať, ukazovateľ na objekt zariadenia, v ktorom sa filtre vytvárajú a vracanú hodnotu identifikátora popisu, slúžiaceho na zrušenie registrácie pri ukončovaní behu ovládača. Návrátová hodnota oznamuje, či bol beh funkcie úspešný. Štruktúra funkcie je zobrazená na obr. 4.5.



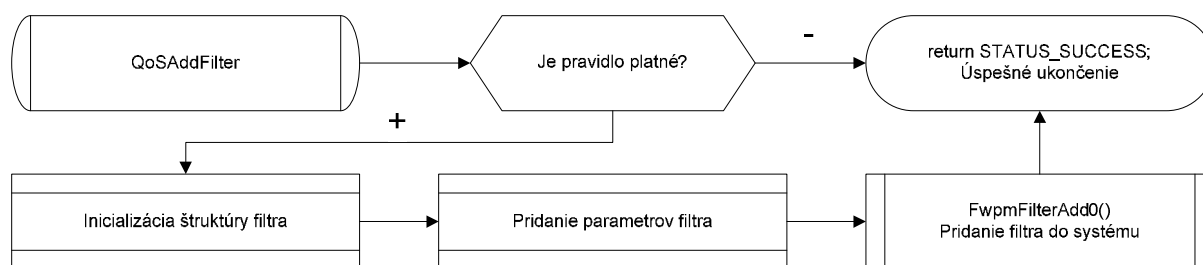
Obr. 4.5 - Štruktúra funkcie pre registráciu popisných ovládačov pre transportnú vrstvu QoSAddFilter()

```

NTSTATUS QoSAddFilter(IN const GUID *layerKey, IN const GUID *calloutKey, IN
OUT QOS_RULE *rule)
  
```

Táto funkcia slúži k samotnej príprave a registrácii filtrov do základného stroja pre filtrovanie (angl. „base filtering engine“). Na začiatku sa inicializuje štruktúra filtra a nastaví sa tak, že filter bude zachytenú premávku blokovať. Následne sú do filtra pridané jednotlivé parametre podľa ich hodnôt v zodpovedajúcom pravidle. Na koniec je filter pridaný do základného stroja pre filtrovanie volaním funkcie FwpmFilterAdd0(), pričom identifikátor filtra je uložený pre neskoršie porovnávanie pri značkovaní.

Parametrami funkcie sú GUID vrstvy, GUID popisu a ukazovateľ na štruktúru obsahujúcu pravidlo, ktorého parametre je potrebné pridať ako parametre filtra. Štruktúra funkcie je zobrazená na obr. 4.6.

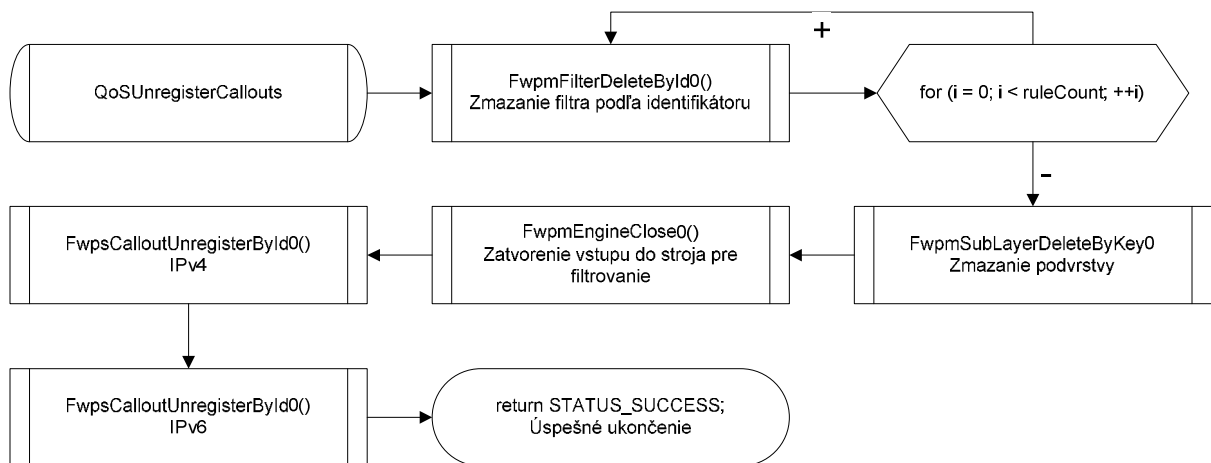


Obr. 4.6 - Funkcia pre registráciu filtrov

QoSUnregisterCallouts()

NTSTATUS QoSUnregisterCallouts()

Funkcia slúži kzrušeniu registrácie popisu, či už pri chybe priamo z funkcie pre inicializáciu, alebo z ukončovacej funkcie. Funkcia najprv odstráni zaregistrované filtre volaním funkcie FwpmFilterDeleteById0(), ďalej je odstránená podvrstva funkciou FwpmSubLayerDeleteByKey0(), uzatvorený prístup do stroja pre filtrovanie volaním funkcie FwpmEngineClose0() a na koniec zrušená registrácia transportných popisov funkciou FwpsCalloutUnregisterById0(). Funkcia nepotrebuje žiadne parametre, jej štruktúra je zobrazená na obr. 4.7.



Obr. 4.7 - Štruktúra funkcie pre zrušenie registrácie popisu

4.2.3 Filtračná logika ovládača

QoSClassify()

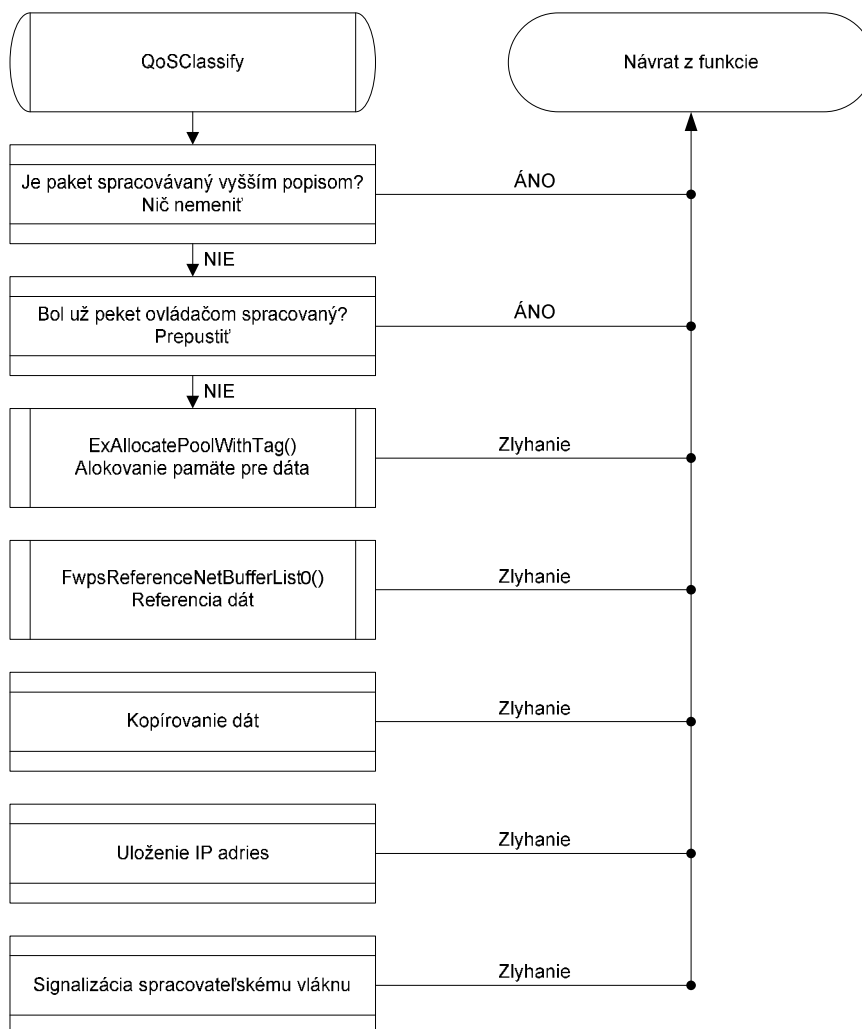
```
VOID QoSClassify(IN const FWPS_INCOMING_VALUES0* inFixedValues, IN const
FWPS_INCOMING_METADATA_VALUES0* inMetaValues, IN OUT void* layerData, IN
const FWPS_FILTER0* filter, IN UINT64 flowContext, OUT
FWPS_CLASSIFY_OUT0* classifyOut);
```

Funkcia pre klasifikáciu je volaná strojom pre filtrovanie zakaždým, keď sieťová premávka je zachytená filtrom zaregistrovaným ovládačom. Funkcia na začiatku overí, či práve zachytené dáta už neboli ovládačom spracované, ak áno, tak ich ihneď povolí. Tiež je vykonaná kontrola, či dané dáta nespracováva nadradený popisný ovládač. Ak áno, tak je paket ignorovaný.⁴⁷ Ďalej sú uložené dáta a meta dáta potrebné pre označkovanie paketu. A po pripravení týchto dát je predaný signál spracovateľskému vláknu. V prípade chyby je paket prepustený, aby nedošlo k strate dát.

Parametrami funkcie sú ukazovateľ na štruktúru obsahujúcu dáta o spracovávanej sieťovej premávke, v tomto prípade dátové štruktúry transportnej vrstvy, ukazovateľ na štruktúru obsahujúcu dáta o dátach spracovávanej sieťovej premávky, ukazovateľ na dáta

⁴⁷ bude filtrom pravdepodobne zachytený neskôr, kedy bude môcť byť spracovaný

vrstvy, ukazovateľ na štruktúru filtra, identifikátor kontextu toku,⁴⁸ a spätný parameter identifikujúci akciu, ktorá má byť nad dátami vykonaná. Štruktúra funkcie pre kvalifikáciu je zobrazená na obr. 4.8.



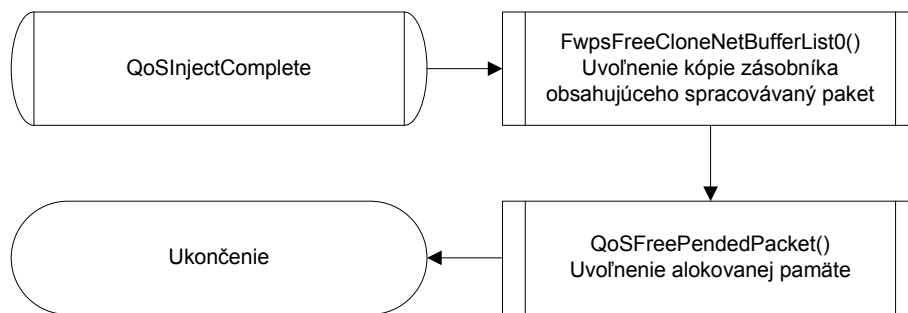
Obr. 4.8 - Štruktúra funkcie pre klasifikáciu

QoSInjectComplete()

```
VOID NTAPI QoSInjectComplete(IN PVOID context, IN OUT NET_BUFFER_LIST*
    netBufferList, IN BOOLEAN dispatchLevel);
```

Funkcia má jednoduchú úlohu, a to uvoľniť alokované prostriedky využitím systémovej funkcie FwpsFreeCloneNetBufferList() a pomocnej funkcie QoSFreePendedPacket(). Funkcia je volaná strojom pre filtrovanie po úspešnom vložení paketu do sieťového zásobníka. Ukazovatele na alokované prostriedky sú odovzdané ako parametre funkcie. Štruktúra funkcie je zobrazená na obr. 4.9.

⁴⁸ Iba pre vrstvy podporujúce toky



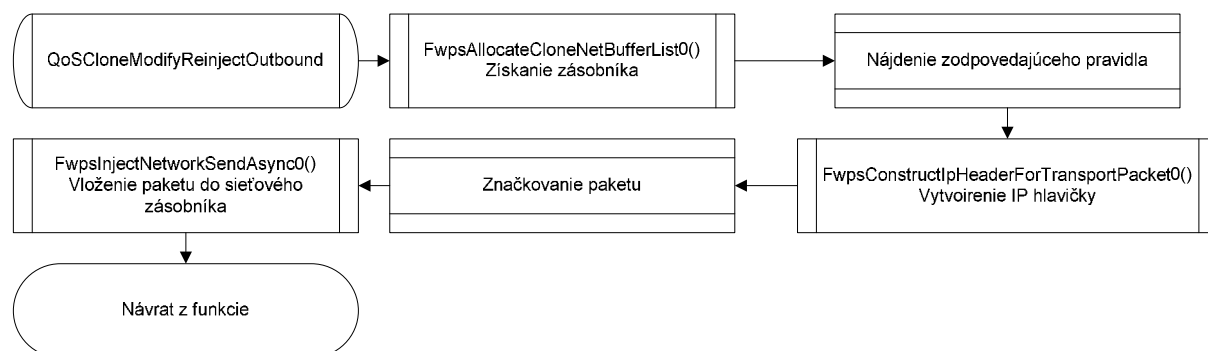
Obr. 4.9 - Štruktúra funkcie pre dokončenie vloženia paketu

QoSCloneModifyReinjectOutbound()

```
NTSTATUS QoSCloneModifyReinjectOutbound(IN QOS_PENDEDED_PACKET* packet);
```

Funkcia slúži k samotnému značkovaniu paketov. Je volaná zo spracovateľského vlákna pre každý paket. Má jediný parameter, a to ukazovateľ na štruktúru obsahujúcu dáta o spracovávanej sieťovej premávke. Štruktúra funkcie je zobrazená na obr. 4.10.

Funkcia na začiatku volá `FwpsAllocateCloneNetBufferList()` pre získanie zásobníka vytvoreného vo funkcii pre klasifikáciu. Následne je vyhľadane pravidlo, ktoré zachytilo dáta pre spracovanie. Ďalej je pre tieto dáta vytvorená IP hlavička pomocou volania systémovej funkcie `FwpsConstructIpHeaderForTransportPacket0()`. Následne je vykonané vlastné značkovanie a na koniec je paket vložený do sieťovej vrstvy sieťového zásobníka volaním funkcie `FwpsInjectNetworkSendAsync0()`.



Obr. 4.10 - Štruktúra funkcie pre značkovanie paketov

QoSNotify()

```
NTSTATUS QoSNotify(IN FWPS_CALLOUT_NOTIFY_TYPE notifyType, IN const GUID *filterKey, IN const FWPS_FILTER0 *filter);
```

Funkcia slúži na informovanie ovládača o udalostiach. Nevyužíva sa, len musí byť implementovaná, pretože stroj pre filtrovanie vyžaduje jej volanie.

QoSFreePendedPacket()

```
__inline VOID QoSFreePendedPacket(IN OUT QOS_PENDEDED_PACKET* packet);
```

Funkcia slúži na uvoľnenie prostriedkov vyhradených pre prácu s dátami a vloženie paketu do sieťového zásobníka. Na začiatku zníži počet referencií na dáta volaním funkcie

FwpsDereferenceNetBufferList0(), následne volaním funkcie ExFreePoolWithTag() uvoľní najprv pamäť pre riadiace dáta a následne aj pamäť pre dáta pre spracovanie premávky.

QoSWorker()

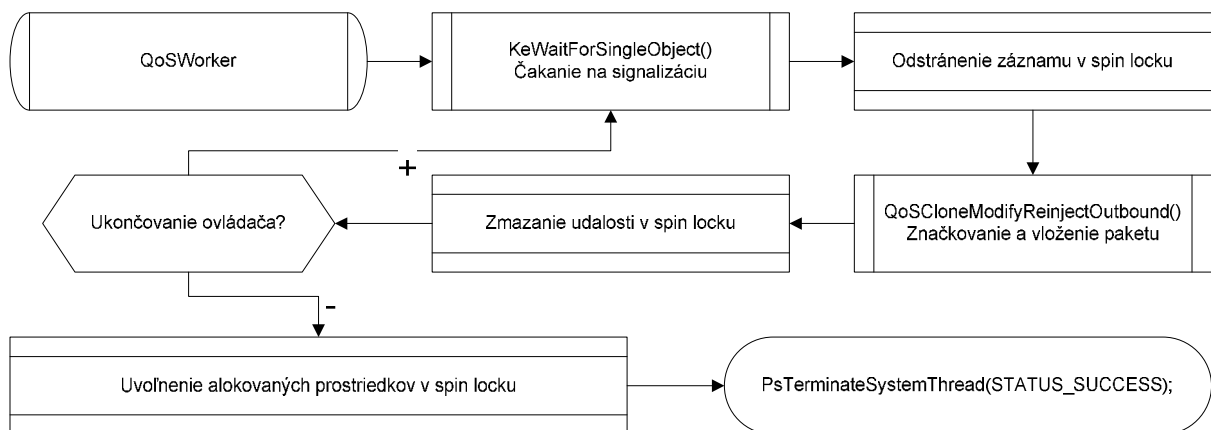
```
VOID QoSWorker(IN PVOID StartContext);
```

Funkcia tvorí telo spracovateľského vlákna a nie je volaná ako bežná funkcia, je z nej vytvorené nové vlákno. Tomu je prispôsobený predpis funkcie. Štruktúra spracovateľského vlákna je zobrazená na obr. 4.11.

Spracovanie dát sa deje v nekonečnej slučke, z ktorej sa vystúpi iba v prípade ukončovania behu ovládača. V prípade, že sú spracované všetky dáta v zozname, vlákno sa zablokuje a čaká na signalizáciu ďalších dát. Inak pokračuje v behu, kým nespracuje všetky dáta.

Na začiatku je volaná funkcia pre odstránenie prvého záznamu zo zoznamu pre spracovanie a tento záznam je následne spracovávaný. Je preň zavolaná funkcia QoSCloneModifyReinjectOutbound() pre označkovanie a vloženie do sieťového zásobníka. Ak operácia prebehne v poriadku, je zodpovednosť za prostriedky predaná stroju pre filtrovanie, inak je potrebné alokované prostriedky uvoľniť volaním pomocnej funkcie QoSFreePendedPacket(). Ďalej nasleduje zistenie, či je zoznam záznamov prázdny. Ak áno, je zrušená signalizácia potreby spracovania dát a v ďalšej iterácii sa beh zablokuje na čakaní na tento signál. Informácie o zamykaní (angl. „spin lock“) a synchronizácii je možné nájsť v [3].

Ak bol vyslaný signál pre ukončenie behu ovládača, tak vlákno vystúpi z nekonečnej slučky a uvoľní alokované prostriedky. Na koniec je beh vlákna ukončený volaním funkcie PsTerminateSystemThread().



Obr. 4.11 - Štruktúra spracovateľského vlákna

4.3 Aplikačné rozhranie ovládača

Aplikačné rozhranie ovládača bolo navrhnuté a implementované na základe špecifikácie načrtnutej v kap. 3.5. Poskytuje sadu funkcií pokrývajúcich celú škálu možností ovládača.

Tieto sú popísané v kap. 4.3.1. Volanie funkcií je možné z prakticky ľubovoľného jazyka.⁴⁹ V jazyku C a C++ je možné využiť nasledujúce metódy:

- **dynamické linkovanie pri štarte aplikácie** – využíva poskytnutý hlavičkový súbor a pri linkovaní aplikácie k nej pripojí malú statickú knižnicu obsahujúcu informácie o tom, ako prilinkovať knižnicu a získať adresy funkcií z nej pri štarte aplikácie, má mierny dopad na rýchlosť štartu aplikácie⁵⁰ a je jednoduchšie na použitie;
- **dynamické linkovanie počas behu aplikácie** – využíva funkcie OS pre zavedenie a dynamické prilinkovanie knižnice k programu počas jeho behu, pre použitie tohto spôsobu je potrebná len samotná DLL knižnica a znalosť predpisov funkcií API, keďže načítanie knižnice je posunuté až na neskôr, čas načítania aplikácie je kratší, táto metóda je zároveň mierne zložitejšia na použitie, ale umožňuje načítať rôzne varianty knižníc podľa potreby.⁵¹ Podrobnejšie informácie o použití je možné nájsť v [24] a v [25].

4.3.1 Funkcie aplikačného rozhrania pre riadenie ovládača

Aplikačné rozhranie ovládača pokrýva plný rozsah možností ovládača a tiež poskytuje pomocné funkcie pre zjednodušenie tvorby nadradených aplikácií a zvýšenie robustnosti systému ako celku. Poskytované funkcie je možné rozdeliť do niekoľkých kategórií:

- funkcie pre načítanie a uloženie pravidiel;
- funkcie pre prácu s pravidlami;
- funkcie pre riadenie ovládača;
- funkcie pre získanie hodnôt pravidiel;
- funkcie pre spracovanie používateľských dát;
- funkcie pre získanie štandardných hodnôt parametrov.

Aplikačné rozhranie je navrhnuté s prihliadnutím na jednoduchosť a robustnosť, a preto funkcie aplikačného rozhrania vykonávajú svoju úlohu v každom prípade, keď je to možné. Volanie určitých funkcií nemá opodstatnenie v určitom kontexte, napr. volanie funkcií pre riadenie ovládača v prípade, že ovládač v systéme nie je nainštalovaný. V takom prípade volaná funkcia nevykoná žiadnu činnosť. Udržiavanie informácie o tomto kontexte je ponechané na aplikácii. Ďalším príkladom je prepísanie neuložených pravidiel v pamäti pri načítaní novej sady pravidiel zo systémových registrov alebo z REG súboru.

⁴⁹ jedinou požiadavkou je podpora volania funkcií z DLL knižníc, čo v podstate podporuje každý programovací jazyk na platforme Windows

⁵⁰ dopad sa zvyšuje s rastúcim počtom takto využívaných DLL knižníc

⁵¹ napr. rôzne jazykové mutácie, alebo rôzne komponenty implementujúce rovnaké rozhranie, atď.; pre prácu s aplikačným rozhraním ovládača pre filtrovanie sieťovej premávky táto vlastnosť využitie nemá

4.3.2 Funkcie pre načítanie a uloženie pravidiel

Táto skupina funkcií poskytuje prácu s úložiskami pravidiel. Aplikačné rozhranie podporuje prácu s dvoma typmi úložísk pravidiel:

- **registre operačného systému** – používajú sa v prípade, že aplikačné rozhranie je volané na systéme, na ktorom je nasadený ovládač, táto možnosť umožňuje správu pravidiel na lokálnom systéme;
- **REG súbory** – používajú sa v prípade distribúcie pravidiel na iné systémy ako lokálny a v prípade udržiavania rôznych verzií či špecifických konfigurácií. REG súbory uložené aplikačným rozhraním sa od REG súborov vyexportovaným z registrov operačného systému líšia v tom, že navyše obsahujú aj položky pre zmazanie nepoužitých parametrov pravidiel [26], čo umožňuje správnu funkciu v prípade prepisu pravidiel na cieľovom systéme bez nutnosti externého zásahu.

LoadRulesFromSystem()

```
void LoadRulesFromSystem(void);
```

Funkcia slúži na načítanie pravidiel z registrov operačného systému. Načítané pravidlá nahradia pravidlá aktuálne sa nachádzajúce v pamäti. Pravidlá sú pri načítavaní kontrolované a do zoznamu pravidiel sú pridané iba platné pravidlá.⁵² V prípade jednoduchých logických chýb sú tieto opravené.

LoadRulesFromFile()

```
void LoadRulesFromFile(wchar_t* filename);
```

Funkcia slúži na načítanie pravidiel z REG súboru, ukazovateľ na reťazec obsahujúci názov tohto súboru je parametrom funkcie. Načítané pravidlá nahradia pravidlá aktuálne sa nachádzajúce v pamäti. Pravidlá sú pri načítavaní kontrolované a do zoznamu pravidiel sú pridané iba platné pravidlá.

SaveRulesToSystem()

```
void SaveRulesToSystem(void);
```

Funkcia slúži na uloženie pravidiel do registrov operačného systému. Prípadné existujúce pravidlá sú prepísané.

SaveRulesToFile()

```
void SaveRulesToFile(wchar_t* filename);
```

Funkcia slúži na uloženie pravidiel do REG súboru, ukazovateľ na reťazec obsahujúci názov tohto súboru je parametrom funkcie. V prípade že súbor existuje, je jeho obsah prepísaný.

⁵² za platné pravidlo je považované také, ktoré obsahuje minimálne špecifikáciu DSCP značky

4.3.3 Funkcie pre prácu s pravidlami

Tieto funkcie umožňujú základnú prácu s pravidlami a ich správu v pamäti. Funkcie majúce za parameter poradové číslo pravidla, jeho hodnotu kontrolujú a v prípade neexistujúceho pravidla nie je vykonaná žiadna akcia. Číslovanie pravidiel začína od nuly.

AddRule()

```
void AddRule(wchar_t* destIP, wchar_t* srcIP, UINT16 destPort, UINT16 srcPort,
            UINT8 protocol, UINT8 DSCP);
```

Umožňuje prídanie nového pravidla do zoznamu pravidiel. Pravidlo je pridávané na koniec zoznamu⁵³ a prípadné logické nesúdržnosti sú opravené. Pre spracovanie používateľského vstupu na parametre pre túto funkciu je potrebné použiť zodpovedajúcu funkciu pre spracovanie používateľských dát.

EditRule()

```
void EditRule(UINT32 ruleNumber, wchar_t* destIP, wchar_t* srcIP, UINT16
            destPort, UINT16 srcPort, UINT8 protocol, UINT8 DSCP);
```

Funkcia umožňuje úpravu parametrov už existujúceho pravidla, ktorého poradové číslo je uvedené ako prvý parameter funkcie. Ostatné parametre ako aj správanie funkcie je totožné s funkciou AddRule() pre pridanie pravidla.

EraseRule()

```
void EraseRule(UINT32 ruleNumber);
```

Slúži na odstránenie pravidla, ktorého poradové číslo je uvedené ako parameter funkcie. Odstraňovať je možné ľubovoľné pravidlo bez ohľadu na jeho pozíciu v zozname.

GetRuleCount()

```
UINT32 GetRuleCount(void);
```

Funkcia vracia aktuálny počet pravidiel v zozname pravidiel.

ClearRules()

```
void ClearRules(void);
```

Funkcia vymazáva zoznam pravidiel a teda z neho odstraňuje všetky pravidlá.

PromoteRule()

```
void PromoteRule(UINT32 ruleNumber);
```

Funkcia slúži k zmene priority pravidiel, konkrétne k zvýšeniu priority pravidla, ktorého poradové číslo je zadané ako parameter funkcie o jedna. V prípade, že parametrom je prvé alebo neexistujúce pravidlo, nevykoná sa žiadna akcia.

⁵³ pre zmenu poradia pravidiel je možné využiť funkcie PromoteRule() a DemoteRule()

DemoteRule()

```
void DemoteRule(UINT32 ruleNumber);
```

Funkcia slúži k zmene priority pravidiel, konkrétne k zníženiu priority pravidla, ktorého poradové číslo je zadané ako parameter funkcie o jedna. V prípade, že parametrom je posledné alebo neexistujúce pravidlo, nevykoná sa žiadna akcia.

4.3.4 Funkcie pre riadenie ovládača

Tieto funkcie umožňujú získanie aktuálneho stavu a riadenie behu ovládača. Použitie riadiacich funkcií má význam len v prípade, že na lokálnom systéme je ovládač nainštalovaný, v opačnom prípade nevykonajú žiadnu činnosť.

QosMarkingStart()

```
void QosMarkingStart(void);
```

Funkcia slúži pre spustenie behu ovládača a teda začiatku filtrovania sieťovej premávky a značkovania paketov DSCP značkami podľa pravidiel uložených v registroch operačného systému.

QosMarkingStop()

```
void QosMarkingStop(void);
```

Funkcia slúži na zastavenie behu ovládača a teda ukončenia filtrovania sieťovej premávky a značkovania paketov DSCP značkami podľa pravidiel uložených v registroch operačného systému.

QosMarkingRestart()

```
void QosMarkingRestart(void);
```

Funkcia slúži k reštartovaniu behu ovládača a teda k novému načítaniu pravidiel pre filtrovanie sieťovej premávky a značkovanie paketov DSCP značkami z registrov operačného systému.

DriverStatus()

```
UINT8 DriverStatus(void);
```

Funkcia slúži k zisteniu aktuálneho stavu ovládača na lokálnom systéme. Návratová hodnota môže byť 0 v prípade, že ovládač beží, 1 v prípade, že ovládač nebeží a 2, ak ovládač na lokálnom systéme nie je nainštalovaný.

4.3.5 Funkcie pre získanie hodnôt parametrov pravidiel

Funkcie z tejto kategórie umožňujú získanie hodnôt parametrov pravidiel zo zoznamu pravidiel. Každá funkcia vracia ukazovateľ na textovú reprezentáciu hodnoty určitého parametra pravidla, ktorého poradové číslo je uvedené ako parameter funkcie. V prípade neexistujúceho pravidla je vrátený prázdny ukazovateľ (angl. „NULL pointer“). V prípade, že hodnota sa nachádza medzi štandardnými hodnotami, je vrátený ukazovateľ na textovú

reprezentáciu zhodnú s tou, ktorá je využívaná funkciami pre získanie zoznamu štandardných hodnôt.

GetDestinationIP()

```
wchar_t* GetDestinationIP(UINT32 ruleNumber);
```

Vracia ukazovateľ na textovú reprezentáciu cieľovej IP adresy zadaného pravidla. IP adresa môže byť typu IPv4 alebo IPv6 s prípadnou špecifikáciou dĺžky sieťového prefixu za znakom delenia.

GetSourceIP()

```
wchar_t* GetSourceIP(UINT32 ruleNumber);
```

Vracia ukazovateľ na textovú reprezentáciu zdrojovej IP adresy zadaného pravidla. IP adresa môže byť typu IPv4 alebo IPv6 s prípadnou špecifikáciou dĺžky sieťového prefixu za znakom delenia.

GetDestinationPort()

```
wchar_t* GetDestinationPort(UINT32 ruleNumber);
```

Vracia ukazovateľ na textovú reprezentáciu cieľového portu transportnej vrstvy.

GetSourcePort()

```
wchar_t* GetSourcePort(UINT32 ruleNumber);
```

Vracia ukazovateľ na textovú reprezentáciu zdrojového portu transportnej vrstvy.

GetProtocol()

```
wchar_t* GetProtocol(UINT32 ruleNumber);
```

Vracia ukazovateľ na textovú reprezentáciu cieľového protokolu transportnej vrstvy.

GetDSCP()

```
wchar_t* GetDSCP(UINT32 ruleNumber);
```

Vracia ukazovateľ na textovú reprezentáciu DSCP značky. Táto značka má binárny tvar.

4.3.6 Funkcie pre spracovanie používateľských dát

Funkcie z tejto skupiny slúžia na spracovanie používateľského vstupu a získanie hodnôt parametrov. Tieto funkcie je nutné používať namiesto funkcií pre načítavanie hodnôt z textových reťazcov poskytovaných štandardnou knižnicou použitého jazyka alebo aplikačným rozhraním operačného systému. V prípade neplatného vstupu funkcie vracajú predvolenú hodnotu pre konkrétny parameter.

ConvertPorttoNumber()

```
UINT16 ConvertPorttoNumber(wchar_t* port);
```

Funkcia konvertuje reťazec, ukazovateľ na ktorý je funkcii predaný ako parameter, na numerickú reprezentáciu portu⁵⁴ transportnej vrstvy, ktorá je návratovou hodnotou funkcie.

ConvertProtocoltoNumber()

```
UINT8 ConvertProtocoltoNumber(wchar_t* protocol);
```

Funkcia konvertuje reťazec, ukazovateľ na ktorý je funkcii predaný ako parameter, na numerickú reprezentáciu protokolu transportnej vrstvy, ktorá je návratovou hodnotou funkcie.

ConvertDSCPtoNumber()

```
UINT8 ConvertDSCPtoNumber(wchar_t* DSCP);
```

Funkcia konvertuje reťazec, ukazovateľ na ktorý je funkcii predaný ako parameter, na numerickú reprezentáciu DSCP značky, ktorá je návratovou hodnotou funkcie.

4.3.7 Funkcie pre získanie štandardných hodnôt parametrov

Tieto funkcie umožňujú získanie zoznamu často používaných či významnejších hodnôt parametrov pravidiel. Pre každý parameter existuje funkcia na získanie počtu a ďalšia na získanie hodnoty s konkrétnym indexom. Číslovanie hodnôt sa začína nulou a v prípade neplatného indexu je vrátený ukazovateľ na prázdny reťazec.

GetPortTypeCount()

```
UINT32 GetPortTypeCount(void);
```

Funkcia vracia počet štandardných hodnôt pre parametre zdrojový port a cieľový port.

GetProtocolTypeCount()

```
UINT32 GetProtocolTypeCount(void);
```

Funkcia vracia počet štandardných hodnôt parametru protokolu transportnej vrstvy.

GetDSCPTypeCount()

```
UINT32 GetDSCPTypeCount(void);
```

Funkcia vracia počet štandardných hodnôt parametru DSCP značky.

GetPortTypeByID()

```
wchar_t* GetPortTypeByID(UINT32 number);
```

Funkcia vracia textovú reprezentáciu zdrojového alebo cieľového portu s indexom zadaným ako parameter.

⁵⁴ bez ohľadu na to či ide o zdrojový alebo cieľový port

GetProtocolTypeID()

```
wchar_t* GetProtocolTypeID(UINT32 number);
```

Funkcia vracia textovú reprezentáciu protokolu transportnej vrstvy s indexom zadaným ako parameter.

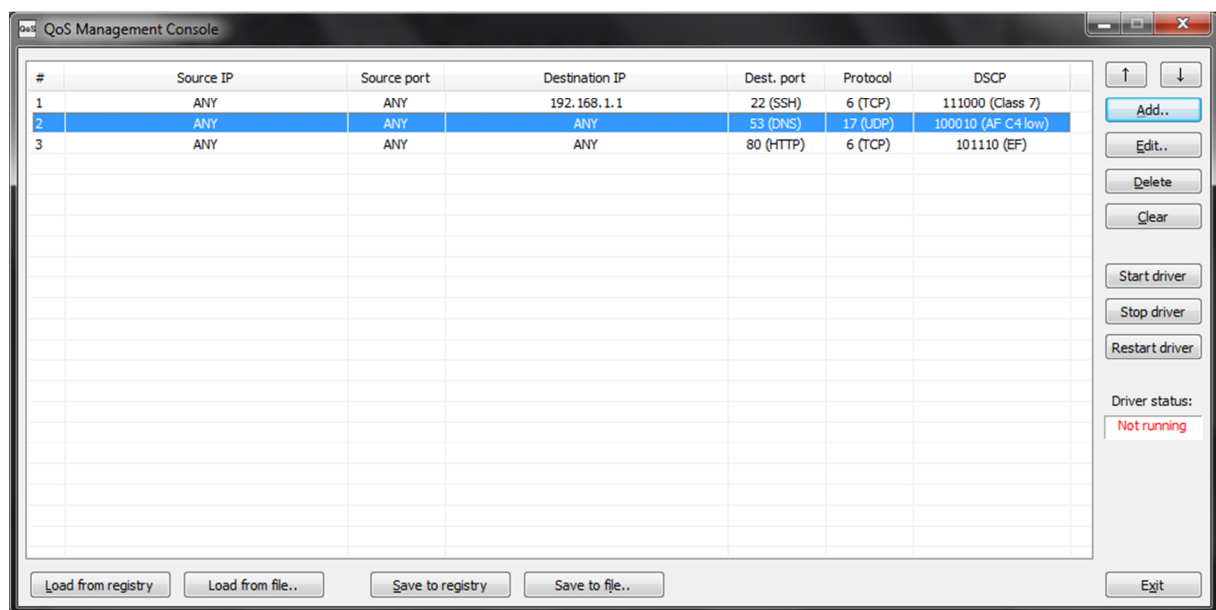
GetDSCPTypeID()

```
wchar_t* GetDSCPTypeID(UINT32 number);
```

Funkcia vracia textovú reprezentáciu DSCP značky s indexom zadaným ako parameter.

4.3.8 QoS Management Console

Pre zjednodušenie použitia bol súčasťou práce návrh a dizajn aplikácie pre správu pravidiel a riadenie ovládača. Táto aplikácia plne využíva možnosti API a je preň v podstate iba grafickým užívateľským rozhraním. Hlavné okno aplikácie je zobrazené na obr. 4.12. Jedná sa o jednoduchý formulár, obsahujúci zoznam pravidiel a niekoľko tlačidiel na prácu s ním a so stavom samotného ovládača. Aplikácia má jedno sekundárne okno slúžiace na pridávanie a editáciu pravidiel.



Obr. 4.12 - Hlavné okno QoS Management Console

Prvé dve tlačidlá (s obrázkami šípok) slúžia na zmenu poradia pravidiel. Poradie pravidiel ovplyvňuje ich uplatnenie pri filtrovaní, pretože každý paket je značkovaný podľa prvého pravidla, ktoré spĺňa. Ďalšie dve tlačidlá slúžia na pridávanie nových alebo editáciu už existujúcich pravidiel. Okno pre pridávanie pravidiel je zobrazené na obr. 4.13. Okno pre úpravu pravidiel je takmer identické (technicky sa jedná o jeden dialógový zdroj). Zdrojová a cieľová IP adresa môžu byť IPv4 alebo IPv6 podľa zaužívaných pravidiel zápisu (IPv4 používa dekadický zápis, oddeľovačom je bodka, sú povinné všetky štyri čísla; IPv6 používa hexadecimálny zápis, oddeľovačom je dvojbodka, je možný skrátenejší zápis pomocou dvoch dvojbodiek ako je pri IPv6 adresách zvykom) a s pridaním dĺžky sieťového prefixu za znakom

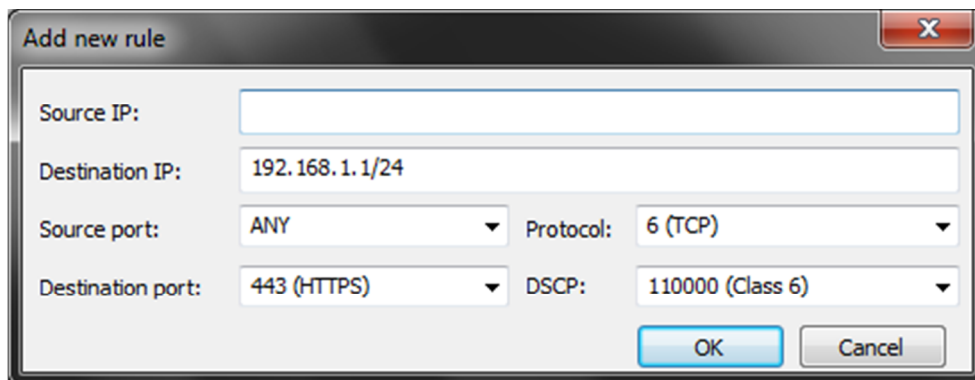
delenia na konci adresy. V prípade pridania dĺžky sieťového segmentu je ako parameter pravidla použitá celá takto špecifikovaná podsieť. V prípade, že políčko pre zadanie IP adresy nebude vyplnené, pravidlo nebude filtrovať podľa IP adresy.

Zvyšné položky – zdrojový a cieľový port, protokol a DSCP značku je možné vyberať z vysúvacieho zoznamu, alebo zadávať numericky. Zdrojový port, cieľový port a protokol sa zapisujú dekadicky, DSCP značka v binárnom tvare.

V prípade zadania chybného či nezmyselného údaju do niektorého zo vstupných polí sa táto hodnota ako parameter filtra nepoužije.

Tlačidlo „Delete“ umožňuje zmazať aktuálne vybraný záznam, zatiaľ čo tlačidlo „Clear“ vymaže všetky záznamy. Ďalšie tri tlačidlá slúžia na prácu so stavom ovládača. Je možné ho spúšťať, zastavovať a reštartovať. Aktuálny stav ovládača je zobrazený na ukazovateli pod tlačidlami na ovládanie stavu. Ukazovateľ je aktualizovaný po zmene stavu ovládača z programu, ale môže byť aktualizovaný aj na používateľský pokyn kliknutím na ukazovateľ.

Posledná skupina tlačidiel, nachádzajúca sa pod zoznamom pravidiel umožňuje načítanie a ukladanie pravidiel. Pravidlá je možné ukladať a načítavať z registrov OS v prípade použitia na lokálnom počítači, alebo je možné pracovať s REG súbormi pre distribúciu pravidiel na iné systémy. Práca s REG súbormi je výhodná najmä v prípade rovnakého nastavenia viacerých staníc.



Obr. 4.13 - Formulár pre pridanie nového pravidla

Aplikácia sa stará o stav dát, takže pred operáciami prepisujúce dáta bez ich predchádzajúceho uloženia sa spýta na potvrdenie.

Aplikácia je vytvorená s použitím knižnice MFC, informácie o využití tejto knižnice na tvorbu aplikácií boli čerpané z [19] a [20], kde je možné o nej nájsť podrobné informácie.

5 Záver

Práca si dávala za úlohu oboznámenie s funkciou ovládačov v OS rodiny Windows a princípmi ich tvorby so zameraním na sieťové ovládače.

Cieľom praktickej časti práce bola tvorba ovládača pre značkovanie sieťovej premávky DSCP značkami a pomocných programov. Praktickým výstupom práce teda je samotný ovládač v režime jadra, umožňujúci nastavovať DSCP značky sieťovej premávke podľa používateľsky definovaných pravidiel. Ďalším dôležitým prvkom je dynamicky linkovaná knižnica v režime používateľa pre manipuláciu s pravidlami a ovládanie ovládača. Túto knižnicu je možné využiť v ďalších vyvíjaných systémoch pre prácu s ovládačom. Pre umožnenie jednoduchšej práce s ovládačom pred vyvinutím ďalších systémov je vytvorená aplikácia s grafickým používateľským rozhraním využívajúca služby poskytované knižnicou. Taktiež bol vytvorený systém pre meranie vplyvu ovládačov pre filtrovanie sieťovej premávky na parametre dátového toku, najmä oneskorenie a priepustnosť.

Teoretická časť obsahuje popis získaných teoretických poznatkov, ale najmä dokumentuje postup tvorby a využitie jednotlivých súčastí praktickej časti. Problematika tvorby ovládačov a programovania pre režim jadra všeobecne je ale komplexná a svojim rozsahom presahuje možnosti tejto práce. Podklady pre štúdium v prípade pokračovania v práci je možné nájsť v použitej literatúre. Zvyšné súčasti praktickej časti sú teoreticky jednoduchšie.

Výstup práce je základná implementácia ovládača pre filtrovanie sieťovej premávky a značkovanie odchádzajúcich paketov podľa definovateľných pravidiel pre zabezpečenie kvality služby (QoS). V čase tvorby neboli presne známe podmienky prevádzkovania riešenia a bolo vychádzané z teoretickým predpokladov. V práci sú popísané jednotlivé možnosti optimalizácie a prispôsobenia reálnym podmienkam. Možným pokračovaním práce je teda prispôsobenie riešenia prevádzkovým požiadavkám, ktoré je možné reálne určiť až v prípade nasadenia.

Použitá literatúra

- [1] Russinovich, Mark E. a Solomon, David A. *Windows Internals, 5th edition*. Redmond : Microsoft Press, 2009. ISBN 978-0-7356-2530-3.
- [2] Russinovich, Mark E. *Windows Internals, 4th edition*. s.l. : Microsoft Press, 2004. 0735619174.
- [3] Microsoft Corporation. *Windows Driver Kit Documentation*. 2009.
- [4] Microsoft Corporation. *Windows NT Hardware Abstraction Layer (HAL)*. *Technická podpora Microsoft*. [Online] 31. október 2006. [Dátum: 17. október 2010.] <http://support.microsoft.com/kb/99588>.
- [5] Microsoft Corporation. *Windows Filtering Platform*. [Online] 11. apríl 2010. [Dátum: 16. december 2010.] <http://www.microsoft.com/whdc/device/network/WFP.mspx>.
- [6] Microsoft Corporation. *Windows Filtering Platform (Windows)*. *MSDN*. [Online] 21. september 2010. [Dátum: 23. október 2010.] [http://msdn.microsoft.com/en-us/library/aa366510\(v=VS.85\).aspx](http://msdn.microsoft.com/en-us/library/aa366510(v=VS.85).aspx).
- [7] Russinovich, Mark. *Pushing the Limits of Windows: Paged and Nonpaged Pool*. *Mark's Blog*. [Online] 26. marec 2009. [Dátum: 21. máj 2011.] <http://blogs.technet.com/b/markrussinovich/archive/2009/03/26/3211216.aspx>.
- [8] Microsoft Corporation. *About the Winsock SPI*. *MSDN*. [Online] 11. apríl 2010. [Dátum: 24. november 2010.] [http://msdn.microsoft.com/en-us/library/ms737522\(v=VS.85\).aspx](http://msdn.microsoft.com/en-us/library/ms737522(v=VS.85).aspx).
- [9] Hua, Wei, Ohlund, Jim a Butterklee, Barry. *Unraveling the Mysteries of Writing a Winsock 2 Layered Service Provider*. *Microsoft Systems Journal*. [Online] máj 1999. [Dátum: 24. november 2010.] <http://www.microsoft.com/msj/0599/layeredservice/layeredservice.aspx>.
- [10] Smirnov, Vadim V. *Firewall for Windows*. *NT Kernel Resources*. [Online] 2000 - 2005. [Dátum: 17. máj 2011.] <http://www.ntkernel.com/w&p.php?id=14>.
- [11] Microsoft Corporation. *A nonpaged pool memory leak occurs when you use a WFP callout driver in Windows Vista, Windows 7, Windows Server 2008, or in Windows Server 2008 R2*. *Technická podpora Microsoft*. [Online] 2010. [Dátum: 21. máj 2011.] <http://support.microsoft.com/kb/979223/>.
- [12] Microsoft Corporation. *Using two Windows Filtering Platform (WFP) drivers causes a computer to crash when the computer is running Windows Vista, Windows 7, or Windows Server 2008*. *Technická podpora Microsoft*. [Online] 2010. [Dátum: 17. máj 2011.] <http://support.microsoft.com/kb/979278/>.
- [13] Microsoft Corporation. *WFP drivers may cause a failure to disconnect the RDP connection to a multiprocessor computer that is running Windows Vista, Windows Server 2008, windows 7 or Windows Server 2008 R2*. *Technická podpora Microsoft*. [Online] 2010. [Dátum: 17. máj 2011.] <http://support.microsoft.com/kb/976759/>.

- [14] Microsoft Corporation. *Windows Filtering Platform (WFP)*. *Windows Developer Center*. [Online] 2011. [Dátum: 17. máj 2011.] <http://social.msdn.microsoft.com/Forums/en-US/wfp/threads>.
- [15] The WinPcap Team. *WinPcap Documentation*. [Online] 2009. [Dátum: 8. december 2010.] http://www.winpcap.org/docs/docs_412/html/main.html.
- [16] NT Kernel Resources. *Windows Packet Filter Kit. Products*. [Online] [Dátum: 23. október 2010.] <http://www.ntkernel.com/w&p.php?id=7>.
- [17] Eckel, Bruce. *Myslíme v C++*. s.l. : Grada Publishing, 2000. ISBN 80-247-9009-2.
- [18] Josuttis, Nicolai M. *C++ Standardní knihovna a STL: Kompletní průvodce*. Brno : CP Books, 2005. ISBN 80-251-0511-1.
- [19] Prociše, Jeff. *Programování ve Windows pomocí MFC*. Praha : Computer Press, 2002. ISBN 80-7226-309-9.
- [20] Kruglinsky, David J.-Shepherd, George a Wingo, Scot. *Programujeme v Microsoft Visual C++*. Praha : Computer Press, 2000. ISBN 80-7226-362-5.
- [21] Dooren, Bruno van. *Designing a Device API: Part I: What It Means, and Why You Should Do It*. *OSR Online*. [Online] NT Insider, 7. február 2007. [Dátum: 21. máj 2011.] <http://www.osronline.com/article.cfm?id=482>.
- [22] Dooren, Bruno van. *Designing a Device API - Part II: Function Declarations*. *OSR Online*. [Online] NT Insider, 6. september 2007. [Dátum: 21. máj 2011.] <http://www.osronline.com/article.cfm?article=500>.
- [23] Dooren, Bruno van. *Designing a Device API Part III: Exporting Functions*. *OSR Online*. [Online] NT Insider, 22. august 2007. [Dátum: 21. máj 2011.] <http://www.osronline.com/article.cfm?article=521>.
- [24] Microsoft Corporation. *What is a DLL? Technická podpora Microsoft*. [Online] 4. december 2007. [Dátum: 23. máj 2011.] <http://support.microsoft.com/kb/815065>.
- [25] Microsoft Corporation. *Using Run-Time Dynamic Linking*. *MSDN*. [Online] 15. december 2010. [Dátum: 23. máj 2011.] [http://msdn.microsoft.com/en-us/library/ms686944\(v=vs.85\).aspx](http://msdn.microsoft.com/en-us/library/ms686944(v=vs.85).aspx).
- [26] Microsoft Corporation. *How to add, modify, or delete registry subkeys and values by using a registration entries (.reg) file*. *Technická podpora Microsoft*. [Online] 3. december 2007. [Dátum: 22. máj 2011.] <http://support.microsoft.com/kb/310516/en-us>.
- [27] Gates, Mark-a iní. *Iperf User Docs*. *NLANR applications support*. [Online] marec 2003. [Dátum: 23. máj 2011.] <http://webfolder.wirelessleiden.nl/iperf/>.
- [28] *D-ITG Documentation*. *D-ITG, Distributed Internet Traffic Generator*. [Online] 2. máj 2008. [Dátum: 23. máj 2011.] <http://www.grid.unina.it/software/ITG/codice/D-ITG2.6.1d-manual.pdf>.
- [29] Wang, Zheng. *Internet QoS: architectures and mechanisms for Quality of Service*. San Francisco : Morgan Kaufmann Publishers, 2001. ISBN 1-55860-608-4.

Zoznam použitých skratiek

ACPI	štandard definujúci jednotný mechanizmus konfigurácie a správy napájania zariadení - Advanced Configuration and Power Interface
ALE	množina vrstiev WFP v režime jadra, používaných pre stavové filtrovanie paketov (SPI) - Application Layer Enforcement
ALPC	forma komunikácie medzi procesmi pre vysokorýchlostné odovzdávanie správ, pokročilé lokálne volanie procedúr - Advanced Local Procedure Call
AMD	výrobca procesorov a integrovaných obvodov - Advanced Micro Devices
API	rozhranie programovania aplikácií - Application Programming Interface
AT&T	najväčší poskytovateľ telefónnych služieb, pripojenia k internetu a služieb digitálnej televízie v Spojených štátoch amerických - American Telephone & Telegraph
AuthIP	proprietárny protokol rozširujúci protokol IKE, vyvinutý firmou Microsoft
BFE	modul riadiaci filtrovanie vo WFP prijímajúci pravidlá, zabezpečujúci bezpečnostný model aplikácie, udržiavajúci štatistické údaje a ukladajúci denník WFP, hlavný stroj pre filtrovanie - Base Filtering Engine
C	procedurálny programovací jazyk pre všeobecné použitie navrhnutý pre jednoduchosť kódu, priame mapovanie konštrukcií na strojový kód a potrebu minimálnej podpory pre beh programov, umožňujúci portabilitu na úrovni zdrojových kódov a použiteľnosť na širokej škále platforiem
C++	objektovo orientovaný programovací jazyk pre všeobecné použitie vychádzajúci z jazyka C, zdieľa väčšinu jeho vlastností a pridáva užitočné vysokoúrovňové konštrukcie a objektový prístup
CAT	prípona katalógových súborov, obsahujúcich rôzne informácie o iných súboroch, voliteľne aj digitálne podpísané - Catalog File
COM	štandard špecifikujúci binárne rozhranie programových komponentov, umožňujúci medziprocesovú komunikáciu a dynamické vytváranie objektov, umožňuje tvorbu komponentov ktoré môžu byť využívané v programovacích jazykoch iných, v akých boli vytvorené – Component Object Model
CoNDIS	súčasť rozhrania NDIS na tvorbu ovládačov pre služby spojovo orientované na linkovej vrstve, spojovo orientované NDIS - Connection-oriented NDIS
DHCP	protokol umožňujúci automatizované priradenie sieťových adries v IPv4 sieťach, protokol pre dynamickú konfiguráciu zariadení - Dynamic Host Configuration Protocol
D-ITG	distribuovaný generátor sieťovej premávky, určený na simuláciu umelej sieťovej premávky majúcej vlastnosti reálnych dátových prenosov, umožňuje meranie parametrov a vyhodnocovanie vplyvu sieťovej infraštruktúry na sieťovú premávku – Distributed Internet Traffic Generator

DIRQL	stupeň IRQL pri ktorom sú povolené iba prerušenia HW, najvyšší stupeň, povolenie prerušenia zariadení - Device Interrupt Request Level
DLL	knižnica obsahujúca kód alebo iné zdroje použiteľné spustiteľnými programami, dynamicky pripájaná za behu programu, používaná na platforme Windows, dynamicky linkovaná knižnica - Dynamic Link Library
DPC	programové prerušenie spúšťané jadrom OS umožňujúce procesu dokončiť prácu pred ukončením alebo zmenou kontextu, odložené volanie procedúr - Deferred Procedure Call
DRM	technológie kontroly prístupu k digitálnemu obsahu a dielam alebo k zariadeniam, správa digitálnych práv - Digital Rights Management
DSCP	pole v hlavičke IP paketu určujúce prioritu paketu pri zabezpečovaní kvality služby variantom diferencovaných služieb, nahrádza staršie pole TOS – Differentiated Services Code Point
GUID	globálne unikátny identifikátor, je vytváraný algoritmom špeciálne navrhnutým tak, aby zabezpečoval tieto vlastnosti – Globally Unique Identifier
HAL	programová vrstva pracujúca priamo s komponentmi počítača a skrývajúca rozdiely v nich pred jadrom OS - Hardware Abstraction Layer
HW	pevné súčasti počítača - Hardware
I/O	vstupno-výstupný - Input / Output
IDS	program pre zisťovanie prieniku neautorizovaných agentov, systém pre detekciu prieniku - Intrusion Detection System
IHV	nezávislý dodávateľ zariadení - Independent Hardware Vendor
IKE	systém spravujúci zdieľanú bezpečnostnú politiku a autorizujúci kľúče pre služby ktoré ich požadujú, výmena kľúčov - Internet Key Exchange
INF	prípona textových súborov používaných v OS Windows pre inštaláciu programov a ovládačov, najčastejšie ovládačov zariadení, súbor s informáciami pre inštaláciu - Setup Information File
IOCTL	príkaz zaslaný zariadeniu kvôli zisteniu alebo zmene správania zariadenia, dát alebo atribútov zariadenia alebo objektov sprístupnených zariadením - I/O Control
IP	protokol sieťovej vrstvy v protokolovej sade TCP/IP, internetový protokol - Internet Protocol
IPERF	nástroj na meranie priepustnosti medzi dvoma bodmi v dátovej sieti
IPSec	súhrn otvorených štandardov zaoberajúcich sa dôvernosťou, integritou a autorizáciou dát medzi spolupracujúcimi entitami na sieťovej vrstve protokolovej sady TCP/IP - IP Security
IPv4	internetový protokol verzie 4 - Internet Protocol version 4
IPv6	internetový protokol verzie 6 - Internet Protocol version 6
IRP	štruktúry režimu jadra používané ovládačmi OS Windows na vzájomnú komunikáciu a komunikáciu s jadrom OS, popisujú I/O požiadavky, paket I/O požiadavky - I/O Request Packet

IRQL	priorita požiadavky na prerušenie, používané pri maskovaní prerušení, stupeň povolenia prerušenia - Interrupt Request Level
ISA	počítačová zbernica IBM PC a kompatibilných - Industry Standard Architecture
ISR	procedúra spätného volania spúšťaná prijatím prerušenia, slúžiaca na spracovanie prerušenia, správca prerušenia - Interrupt Service Routine
ISV	nezávislý dodávateľ aplikácií - Independent Software Vendor
KM	režim jadra - Kernel Mode
KMDF	knižnica pre tvorbu ovládačov pracujúcich v režime jadra - Kernel-Mode Driver Framework
LAN	počítačová sieť spájajúca zariadenia v menšej geografickej oblasti, lokálna počítačová sieť - Local Area Network
FIFO	druh fronty, z ktorej sú požiadavky vyberané v poradí, v akom prišli, prvý prichádzajúci – First In First Out
LPC	služba pre jednoduché predávanie správ medzi procesmi na jednom počítači poskytovaná jadrom Windows NT, využívaná najmä pre komunikáciu interných subsystémov, lokálne volanie procedúr - Local Procedure Call
LSP	dynamicky pripájaná knižnica využívajúca Winsock API pre vloženie samej seba do sieťového zásobníka a následný prístup k sieťovej premávke, poskytovateľ vrstvových služieb - Layered Service Provider
MAC	nižšia podvrstva linkovej vrstvy sieťového zásobníka zodpovedná za prístup k prenosovému médiu - Medium Access Control
MFC	objektovo orientovaná knižnica od spoločnosti Microsoft určená pre tvorbu aplikácií v programovacom jazyku C++, základné triedy spoločnosti Microsoft - Microsoft Foundation Classes
MSDN	úsek spoločnosti Microsoft zodpovedný za správu vzťahov s vývojármi, poskytuje centrálny repozitár dokumentácie k technológiám spoločnosti Microsoft - Microsoft Developer Network
MS-DOS	16-bitový operačný systém spoločnosti Microsoft - Microsoft Disk Operating System
NAT	mechanizmus redukujúci potrebné množstvo IP adries, proces zmeny sieťových adries paketu pri prechode sieťovou infraštruktúrou, preklad sieťových adries - Network Address Translation
.NET	knižnica pre tvorbu aplikácií od spoločnosti Microsoft, podporujúca niekoľko programovacích jazykov, programy vytvorené s jej pomocou bežia vo virtuálnom stroji s automatickým čistením pamäte
NDIS	rozhranie pre programovanie sieťových kariet - Network Driver Interface Specification
OS	operačný systém - Operating System
PCI	počítačová zbernica pre prepojenie komponentov - Peripheral Component Interconnect
PM	systém správy napájania - Power Management

PnP	architektúra automatickej inštalácie a konfigurácie zariadení pod OS Windows - Plug and Play
PPP	štandardný protokol pre zapuzdrenie paketov posielaných cez WAN spojenie a pre spoluprácu sieťových mostov a smerovačov, protokol bod-bod - Point to Point Protocol
QoS	kontrolný mechanizmus rezervácie sieťových zdrojov, schopnosť poskytovať rôzne úrovne priority rôznym dátovým tokom pre zaistenie vlastností prenosovej služby, kvalita služby - Quality of Service
REG	prípona súborov obsahujúcich záznamy pre import do registrov OS – Registry Entries File
RPC	technológia umožňujúca programom vykonať istú svoju časť v inom adresnom priestore, bežne na inom počítači v sieti bez nutnosti programovať špecifiká vzdialeného styku, vzdialené volanie procedúr - Remote Procedure Call
RTOS	operačný systém, v ktorom má každá udalosť nastavenú dobu, v ktorej musí byť obslužená, to umožňuje presné predpovede o maximálnej dobe trvania operácií, operačný systém reálneho času – Real Time Operating System
SCSI	rýchle rozhranie pre pripojenie zariadení - Small Computer Systems Interface
SPI	(Winsock SPI), špecializovaná časť systému Winsock slúžiaca na tvorbu poskytovateľov služieb, napr. sieťových zásobníkov, alebo poskytovateľov menného priestoru, rozhranie poskytovateľov služieb - Service Provider Interface
SPI	mechanizmus používaný v systémoch firewall, slúži na analýzu všetkých prichádzajúcich paketov a prepustenie len tých, ktoré patria k známemu otvorenému spojeniu, stavová analýza paketov - Stateful Packet Inspection
SSH	systém pre kryptograficky bezpečný vzdialený prístup k zariadeniu, poskytuje viaceré služby, napr. prenos súborov a najmä vzdialený terminál – Secure Shell
SYS	prípona súborov obsahujúcich ovládače zariadení OS Windows, systémový súbor - System File
TCP	protokol transportnej vrstvy protokolovej sady TCP/IP zabezpečujúci spojovo orientovanú spoľahlivú službu - Transmission Control Protocol
TCP/IP	množina komunikačných protokolov používaná v Internete a iných dátových sieťach, najmä lokálnych, názov má podľa dvoch najdôležitejších protokolov, TCP a IP, protokolová sada
TDI	protokol pre komunikáciu medzi transportnou vrstvou sieťového zásobníka a jej nadradenou vrstvou v jadre OS Windows, rozhranie ovládača transportnej vrstvy - Transport Driver Interface
TELNET	protokol pre interaktívnu textovú službu vzdialeného prístupu k zariadeniu, vzdialený terminál, kvôli bezpečnostným problémom je nahradzovaný SSH
TOS	pole v hlavičke IP paketu slúžiace na špecifikáciu typu služby a tak nepriamo na určenie správania sa sieťovej infraštruktúry k paketu, dnes nahradené DSCP poľom – Type of Service

UDP	protokol transportnej vrstvy protokolovej sady TCP/IP zabezpečujúci nespoľahlivú nespojovo orientovanú službu - User Datagram Protocol
UNIT8	celočíselný typ bez znamienka o veľkosti 8 bitov – Unsigned Integer 8
UNIT16	celočíselný typ bez znamienka o veľkosti 16 bitov – Unsigned Integer 16
UNIT32	celočíselný typ bez znamienka o veľkosti 32 bitov – Unsigned Integer 32
UM	režim používateľa - User Mode
UMDF	štruktúra pre tvorbu ovládačov pracujúcich v režime používateľa - User-Mode Driver Framework
UNIX	prenositeľný, viac úlohový a viacpoužívateľský operačný systém vyvinutý spoločnosťou AT&T, prípadne rodina operačných systémov odvodených od tohto operačného systému
USB	sériová zbernica pre prepájanie zariadení, má jednoduchý dizajn, poskytuje vysoké prenosové rýchlosti, a podporu PnP - Universal Serial Bus
WAN	počítačová sieť väčšieho geografického rozsahu prepájajúca zariadenia alebo menšie počítačové siete, rozľahlá počítačová sieť - Wide Area Network
WDK	nástroj na tvorbu ovládačov pre OS Windows - Windows Driver Kit
WDM	knižnica pre tvorbu ovládačov pre OS Windows - Windows Driver Model
WHQL	testovací proces spoločnosti Microsoft pre zariadenia a ovládače tretích strán, po úspešnom zvládnutí testov dostáva ovládač digitálny podpis označujúci jeho kompatibilitu, laboratória Windows pre kvalitu zariadení - Windows Hardware Quality Labs
WMI	technológia pre správu systémov založených na technológii Windows, umožňuje tvorbu dávkových kódov monitorujúcich a ovládajúcich zdroje cez sieťové spojenie - Windows Management Instrumentation

Prílohy

Príloha č. 1 - Tabuľka protokolov zapúzdrených v protokole IP	77
Príloha č. 2 – Zoznam dôležitých DSCP tried a ich odporúčaných kódov	80

Príloha č. 1 - Tabuľka protokolov zapúzdrených v protokole IP

Protokol	Popis	Referencie
0	HOPOPT, IPv6 Hop-by-Hop Option	RFC 1883
1	ICMP, Internet Control Message Protocol	RFC 792
2	IGAP, IGMP for user Authentication Protocol IGMP, Internet Group Management Protocol RGMP, Router-port Group Management Protocol	RFC 1112
3	GGP, Gateway to Gateway Protocol	RFC 823
4	Zapuzdrenie IP v IP	RFC 2003
5	ST, Internet Stream Protocol	RFC 1190, RFC 1819
6	TCP, Transmission Control Protocol	RFC 793
7	UCL, CBT	
8	EGP, Exterior Gateway Protocol	RFC 888
9	IGRP, Interior Gateway Routing Protocol	
10	BBN RCC Monitoring	
11	NVP, Network Voice Protocol	RFC 741
12	PUP	
13	ARGUS	
14	EMCON, Emission Control Protocol	
15	XNET, Cross Net Debugger	IEN 158
16	Chaos	
17	UDP, User Datagram Protocol	RFC 768
18	TMux, Transport Multiplexing Protocol	IEN 90
19	DCN Measurement Subsystems	
20	HMP, Host Monitoring Protocol	RFC 869
21	Packet Radio Measurement	
22	XEROX NS IDP	
23	Trunk-1	
24	Trunk-2	
25	Leaf-1	
26	Leaf-2	
27	RDP, Reliable Data Protocol	RFC 908
28	IRTP, Internet Reliable Transaction Protocol	RFC 938
29	ISO Transport Protocol Class 4	RFC 905
30	NETBLT, Network Block Transfer	
31	MFE Network Services Protocol	
32	Medzi uzlový protokol MERIT	
33	DCCP, Datagram Congestion Control Protocol	
34	Third Party Connect Protocol	
35	IDPR, Inter-Domain Policy Routing Protocol	
36	XTP, Xpress Transfer Protocol	
37	Datagram Delivery Protocol	
38	IDPR, Control Message Transport Protocol	

Protokol	Popis	Referencie
39	TP++ Transport Protocol	
40	IL Transport Protocol	
41	IPv6 cez IPv4	RFC 2473
42	SDRP, Source Demand Routing Protocol	
43	IPv6 Routing header	
44	IPv6 Fragment header	
45	IDRP, Inter-Domain Routing Protocol	
46	RSVP, Reservation Protocol	
47	GRE, General Routing Encapsulation	
48	DSR, Dynamic Source Routing Protocol	
49	BNA	
50	ESP, Encapsulating Security Payload	
51	AH, Authentication Header	
52	I-NLSP, Integrated Net Layer Security TUBA	
53	SWIPE, IP so šifrovaním	
54	NARP, NBMA Address Resolution Protocol	
55	Minimal Encapsulation Protocol	
56	TLSP, Transport Layer Security Protocol using Kryptonet key management	
57	SKIP	
58	ICMPv6, Internet Control Message Protocol for IPv6 MLD, Multicast Listener Discovery	
59	IPv6 No Next Header	
60	IPv6 Destination Options	
61	Ľubovoľný vnútorný protokol zariadenia	
62	CFTP	
63	Ľubovoľný protokol lokálnej siete	
64	SATNET and Backroom EXPAK	
65	Kryptolan	
66	MIT Remote Virtual Disk Protocol	
67	Internet Pluribus Packet Core	
68	Ľubovoľný distribuovaný súborový systém	
69	SATNET Monitoring	
70	VISA Protocol	
71	Internet Packet Core Utility	
72	Computer Protocol Network Executive	
73	Computer Protocol Heart Beat	
74	Wang Span Network	
75	Packet Video Protocol	
76	Backroom SATNET Monitoring	
77	SUN ND PROTOCOL-Temporary	
78	WIDEBAND Monitoring	
79	WIDEBAND EXPAK	
80	ISO-IP	

Protokol	Popis	Referencie
81	VMTP, Versatile Message Transaction Protocol	
82	SECURE-VMTP	
83	VINES	
84	TTP	
85	NSFNET-IGP	
86	Dissimilar Gateway Protocol	
87	TCF	
88	EIGRP	
89	OSPF, Open Shortest Path First Routing Protocol MOSPF, Multicast Open Shortest Path First	
90	Sprite RPC Protocol	
91	Locus Address Resolution Protocol	
92	MTP, Multicast Transport Protocol	
93	AX25	
94	IP-within-IP Encapsulation Protocol	
95	Mobile Internetworking Control Protocol	
96	Semaphore Communications Sec Pro	
97	EtherIP	
98	Encapsulation Header	
99	Ľubovoľný vlastný kryptografický protokol	
100	GMTP	
101	IFMP, Ipsilon Flow Management Protocol	
102	PNNI cez IP	
103	PIM, Protocol Independent Multicast	
104	ARIS	
105	SCPS	
106	QNX	
107	Active Networks	
108	IPPCP, IP Payload Compression Protocol	RFC 2393
109	SNP, Sitara Networks Protocol	
110	Compaq Peer Protocol	
111	IPX v IP	
112	VRRP, Virtual Router Redundancy Protocol	RFC 3768, RFC 5798
113	PGM, Pragmatic General Multicast	
114	Ľubovoľný 0-hop protokol	
115	L2TP, Level 2 Tunneling Protocol	
116	DDX, D-II Data Exchange	
117	IATP, Interactive Agent Transfer Protocol	
118	ST, Schedule Transfer	
119	SRP, SpectraLink Radio Protocol	
120	UTI	
121	SMP, Simple Message Protocol	
122	SM	

Protokol	Popis	Referencie
123	PTP, Performance Transparency Protocol	
124	ISIS over IPv4	
125	FIRE	
126	CRTP, Combat Radio Transport Protocol	
127	CRUDP, Combat Radio User Datagram	
128	SSCOPMCE	
129	IPLT	
130	SPS, Secure Packet Shield	
131	PIPE, Private IP Encapsulation within IP	
132	SCTP, Stream Control Transmission Protocol	
133	Fibre Channel	
134	RSVP-E2E-IGNORE	RFC 3175
135	Mobility Header	RFC 3775
136	UDP-Lite, Lightweight User Datagram Protocol	RFC 3828
137	MPLS v IP	RFC 4023
138	MANET protokoly	RFC 5498
139	HIP, Host Identity Protocol	RFC 5201
140	Shim6, Level 3 Multihoming Shim Protocol for IPv6	RFC 5533
141	WESP, Wrapped Encapsulating Security Payload	RFC 5840
142	ROHC, Robust Header Compression	RFC 5858
143 - 252	Nevyužité	
253 254	Vyhradené pre experimenty a testovanie	
255	Rezervované	

Príloha č. 2 – Zoznam dôležitých DSCP tried a ich odporúčaných kódov

Označenie	Typ služby	DSCP	
		binárne	dekadicky
best-effort	bez prioritného zaobchádzania, premávka je obsluhovaná v rovnakom poradí v akom prišla	000000	0
Class 1	najnižšia trieda priority, pre zachovanie kompatibility s TOS	001000	8
AF C1 low	prvá trieda priority AF, najnižšia priorita pre zahadzovanie	001010	10
AF C1 medium	prvá trieda priority AF, stredná priorita pre zahadzovanie	001100	12
AF C1 high	prvá trieda priority AF, najvyššia priorita pre zahadzovanie	001110	14
Class 2	druhá trieda priority, pre zachovanie kompatibility s TOS	010000	16
AF C2 low	druhá trieda priority AF, najnižšia priorita pre zahadzovanie	010010	18
AF C2 medium	druhá trieda priority AF, stredná priorita pre zahadzovanie	010100	20
AF C2 high	druhá trieda priority AF, najvyššia priorita pre zahadzovanie	010110	22
Class 3	tretia trieda priority, pre zachovanie kompatibility s TOS	011000	24

Označenie	Typ služby	DSCP	
		binárne	dekadicky
AF C3 low	tretia trieda priority AF, najnižšia priorita pre zahadzovanie	011010	26
AF C3 medium	tretia trieda priority AF, stredná priorita pre zahadzovanie	011100	28
AF C3 high	tretia trieda priority AF, tretia priorita pre zahadzovanie	011110	30
Class 4	štvrtá trieda priority, pre zachovanie kompatibility s TOS	100000	32
AF C4 low	štvrtá trieda priority AF, najnižšia priorita pre zahadzovanie	100010	34
AF C4 medium	štvrtá trieda priority AF, stredná priorita pre zahadzovanie	100100	36
AF C4 high	štvrtá trieda priority AF, najvyššia priorita pre zahadzovanie	100110	38
Class 5	piata trieda priority, pre zachovanie kompatibility s TOS	101000	40
EF	premávka v tejto triede musí byť za každých okolností odbavená v nastavenom čase, vhodné pre služby s nízkym oneskorením a stratovosťou	101110	46
Class 6	šiesta trieda priority, pre zachovanie kompatibility s TOS	110000	48
Class 7	najvyššia trieda priority, pre zachovanie kompatibility s TOS	111000	56