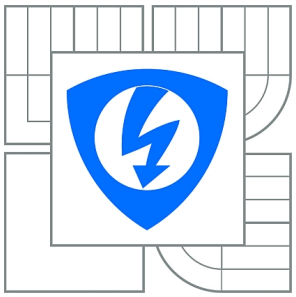
VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ
ÚSTAV TELEKOMUNIKACÍ

FACULTY OF ELECTRICAL ENGINEERING AND COMMUNICATION
DEPARTMENT OF TELECOMMUNICATIONS

MĚŘENÍ PARAMETRŮ SÍŤOVÉHO PROVOZU NA PŘÍSTUPOVÉM BODĚ

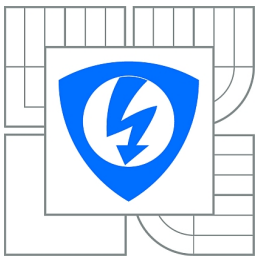
MEASUREMENT OF NETWORK TRAFFIC AT ACCESS POINT

DIPLOMOVÁ PRÁCE
MASTER'S THESIS

AUTOR PRÁCE — Bc. ONDŘEJ NOVÁK
AUTHOR

VEDOUCÍ PRÁCE — doc. Ing. KAROL MOLNÁR, Ph.D.
SUPERVISOR

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. Ondřej Novák ***ID:*** 70348
Ročník: 2 ***Akademický rok:*** 2010/2011

NÁZEV TÉMATU:

Měření parametrů síťového provozu na přístupovém bodě

POKYNY PRO VYPRACOVÁNÍ:

Navrhněte a realizujte aplikaci, která bude monitorovat přístupový bod bezdrátové lokální sítě WLAN a umožňuje pravidelně změřit objem dat přenášených v sestupném i vzestupném směru mezi přístupovým bodem a jednotlivými klientskými stanicemi.
Aplikace dále musí umožnit statistické zpracování získaných dat, např. počítat standardní klouzavý průměr, exponenciální klouzavý průměr, odchylku aktuální hodnoty od klouzavého průměru, najít minimální a maximální hodnoty v určitém intervalu, atd. a výsledné číselné řady znázornit i graficky.

DOPORUČENÁ LITERATURA:

[1] HALLINAN, C.: Embedded Linux Primer: A Practical Real-World Approach, Prentice Hall, 2006
[2] LOVE, R.: Linux Kernel Development, Addison-Wesley Professional, 2010
[3] BOVET, D.P., CESATI, M.: Understanding the Linux Kernel, O'Reilly Media, 2010

Termín zadání: 7.2.2011 ***Termín odevzdání:*** 26.5.2011

Vedoucí práce: doc. Ing. Karol Molnár, Ph.D.

prof. Ing. Kamil Vrba, CSc.
Předseda oborové rady

UPOZORNĚNÍ:

ABSTRAKT

Tato diplomová práce se zaměřuje na problematiku měření parametrů síťového provozu kabelových a bezdrátových sítí. První část teoreticky popisuje protokol SNMP a aplikační rozhraní API, které tvoří základ pro získávání a vyhodnocování dat. Další část popisuje známou platformu MIKROTIK a alternativní softwarové řešení pro možnost implementování nových funkcí. V poslední části je důkladně zdokumentována vytvořená aplikace. Jedná se o jednoduchý informační systém, jehož úkolem je získávat data a staticky je vyhodnocovat pomocí známých funkcí. Závěrem práce je několik testů této aplikace, vyhodnocujících reálný provoz.

Klíčová slova: SNMP, API, Stochastic, Bollinger Bands, Mikrotik, Router OS

ABSTRACT

This master’s thesis focuses on the issue of measurement of parameters of network traffic of wired and wireless networks. The first part theoretically describes SNMP protocol and application interface API, that form the basis for acquiring and evaluating data. The next section describes known MIKROTIK platform and software solutions for the alternative possibility of implementing a new feature. The last part closely describes the created application. It is a simple information system, whose task is to get information and evaluate it using the known functions. The conclusion of the work is a few tests of this application, assessing the real operation.

Keywords: SNMP, API, Stochastic, Bollinger Bands, Mikrotik, Router OS

Prohlášení

Prohlašuji, že svoji diplomovou práci na téma Měření parametrů síťového provozu na přístupovém bodě jsem vypracoval samostatně pod vedením vedoucího semestrálního projektu a s použitím odborné literatury a dalších informačních zdrojů, které jsou všechny citovány v práci a uvedeny v seznamu literatury na konci práce.

Jako autor uvedené diplomové práce dále prohlašuji, že v souvislosti s vytvořením tohoto projektu jsem neporušil autorská práva třetích osob, zejména jsem nezasáhl nedovoleným způsobem do cizích autorských práv osobnostních a jsem si plně vědom následků porušení ustanovení § 11 a následujících autorského zákona č. 121/2000 Sb., včetně možných trestněprávních důsledků vyplývajících z ustanovení § 152 trestního zákona č. 140/1961 Sb.

V Brně dne 20.05.2011 …….

podpis autora

Poděkování

Děkuji vedoucímu diplomové práce doc. Ing. Karolu Molnárovi, Ph.D. za pomoc a rady při zpracování této diplomové práce.

V Brně dne 20.05.2011 podpis autora……..…….

OBSAH

SEZNAM OBRÁZKŮ

ÚVOD

Tato diplomová práce se snaží přiblížit problematiku přístupových bodů majících velký potenciál z hlediska dalšího softwarového rozšíření či modifikace stávajícího chování. V teoretické části je podrobněji pospána dnes velmi rozšířená platforma Mikrotik. Jsou zde zmíněny funkce a typy zařízení RouterBoard a především možnosti konfigurace a správy softwarové části celé platformy RouterOS. Pro porovnání jsou stručně uvedeny vlastnosti volně dostupných softwarových alternativ určené pro běžné přístupové body bezdrátových sítí. Jedná se o systémy OpenWRT a DD-WRT. Dále teoretická část popisuje známý protokol SNMP, který je využíván pro získávání hodnot a parametrů síťového provozu z dnes na trhu dostupných síťových zařízení. V kapitole je popsán způsob vyměňování a typy přenášených informací. Poslední kapitola teoretické části popisuje aplikační programové rozhraní API. Tato kapitola a kapitola popisující protokol SNMP tvoří nutný teoretický základ pro správné pochopení způsobu běhu a fungování aplikace vyvíjené v rámci praktické části tohoto semestrálního projektu.

V praktické části je detailně zdokumentována aplikace s názvem NETMON, která umožňuje měření objemů dat přenesených při komunikaci s bezdrátovými i místními klientskými zařízeními v nastavitelných časových intervalech. Aplikace představuje jednoduché administrátorské prostředí konfigurovatelné z webového prohlížeče. K vývoji aplikace byly použity poznatky o platformě Mikrotik, protokolu SNMP a aplikačním programovém rozhraní API z teoretické části. V dokumentaci jsou uvedeny nejpodstatnější části zdrojových kódů a jejich popis. Výstupem celé aplikace jsou grafy závislosti objemu přenesených dat na době měření v obou směrech. Ty umožňují uživateli porovnat objemy dat mezi přístupovým bodem a více klientskými stanicemi. Dále aplikace umožňuje provádět statistické vyhodnocování pomocí funkcí pro pokročilé zpracování získaných dat. Aplikace generuje krabicové grafy pro možnost delšího monitorování síťového provozu a umožňuje proložit křivky průběhů klouzavým průměrem. Dále jsou v aplikaci implementovány důmyslnější funkce pro zpracování dat jako je oscilátor Stochastic a indikátor volatility Bollinger Bands. Poslední kapitola praktické části diplomové práce se zabývá krátkodobou predikcí intenzity síťového provozu založené na uvedených funkcích statistického vyhodnocování.

1 Protokol SNMP

Protokol SNMP tedy *Simple Network Management Protocol* je standardem v oblasti implementace správy sítě. Jeho základní vlastností je podpora nejrůznějších síťových zařízení a to bez ohledu na architekturu sítě. Vznik protokolu pro jednoduchou správu sítě se datuje do roku 1988 a je součástí sady dokumentů RFC (*Request for Comments*). RFC dalo by se říci, že je celek, definující nejzákladnější principy a možnosti implementace pro protokol, který tvoří standard pro dohled a management sítě Internet. Tento výsledný protokol mohl být aplikován jako univerzální nástroj namísto nesčetných řešení určených pro správu sítě od různých výrobců, které byly v době jeho vzniku dostupné. Postupem času protokol SNMP získával na popularitě, a i navzdory tomu, že jsou dostupné i jiné mechanismy pro správu sítě, stal se SNMP jádrem problematiky správy sítě. Proto u většiny operačních systémů Unix a Windows jsou implementovány agenti SNMP a i z hlediska síťových prvků je SNMP nasazován a implementován do široké škály výrobku od nejrůznějších výrobců. Zejména se jedná od směrovače (routery), mosty (bridge), přepínače (switche), ale i síťové tiskárny a karty. Aplikace agentů SNMP do zmíněných zařízení však může zvyšovat cenu daného zařízení, proto se často setkáváme s verzemi bez SNMP a s implementovaným agentem SNMP.

Postupem času byl protokol SNMP vylepšován a byly opraveny chyby. Původní databáze, známá pod zkratkou MIB (*Management Information Base*), byla doplňována o další položky. MIB standardně definuje síťové objekty v deseti skupinách – systém, rozhraní, NAT, IP, ICMP, TCP, UDP, EGP, data transfer a SNMP. S neustálým technologickým rozvojem dochází k vylepšování zařízení, přidávání nových mechanizmů a technik, výrobci se neustále předhánějí z hlediska výkonu a funkcí, a tak může nastat situace, že standardní objekty a skupiny MIB určité nově přibývající funkce nepokrývají. Aby byly tyto nedostatky alespoň částečně kompenzovány, vyvinuli dodavatelé zařízení s vlastní databází MIB.

Začátkem 90. let byly původní definice protokolu SNMP upraveny a rozšířeny. Byly přidány nové skupiny MIB, některé prvky původní MIB se staly zbytečnými, a proto byly ze specifikace odstraněny. Nová specifikace MIB-2 nahradila původní-první verzi, se kterou je však plně kompatibilní. Následně byla databáze doplněna o možnost vzdáleného monitoringu známou pod zkratkou RMON (*Remote Network Monitoring*). Úkolem tohoto mechanizmu je analyzovat a sledovat stav sítě. Výstupem této funkce jsou informace, které se liší od standardních informací protokolu SNMP. Zjednodušeně řečeno protokol SNMP shromažďuje informace o samotném zařízení a to z pohledu jak jeho funkce, tak vztahu k síti. Naopak mechanizmus RMON shromažďuje informace o provozu v síti a to v obou směrech z pohledu daného zařízení. Jedná se o informace popisující přenosy v síti, historii, databázi hostitelů a informace o nich a další události, ke kterým v síti dochází. Funkcí RMON je nastavení filtrování jen určitého přenosu, v určitém směru týkající se jen požadovaného zařízení.

Postupem času došlo i na otázku zabezpečení. Na základě vyvíjeného tlaku došlo tedy ke vzniku protokolu S-SNMP a to v roce 1992. Je důležité říci, že tento protokol definuje zabezpečovací techniky, avšak s původní verzí SNMP není kompatibilní. Postupem času tedy došlo k doplnění SNMP, a to zejména o funkce zabezpečení, které se objevily v protokolu S-

SNMP. Výsledkem je komplet zvaný SNMPv2. U použití tohoto protokolu však došlo k problémům kompatibility u některých softwarových a hardwarových řešení a navíc pro uživatele nebylo zabezpečení až tak důležité. Proto i v současnosti je mnoho agentů kompatibilních s agenty MIB-2 a nikoli s protokolem SNMPv2.

V roce 1999 vznikla další verze protokolu pod názvem SNMPv3. Tato verze je založena na svém předchůdci SNMPv2 a je doplněna o mapování přenosů, zabezpečení a vzdálenou konfiguraci. [1]

1.1 Manažeři a agenti

Jak už z vlastností protokolu SNMP vyplývá, součástí schématu jsou správce (stanice neboli manažer) a agenti. Stanice je určena pro správu, prohlížení a analýzu, dokonce jejím úkolem je správa síťových zařízení v daném místě. Stanice může být realizována počítačem, či softwarem na počítači. Příkladem může být softwarové řešení SNMP Extension integrované v operačních systémech firmy Microsoft. Pod pojmem agent si můžeme představit aktivní software na spravovaném síťovém prvku, například na směrovači. Jeho úkolem je shromažďovat informace o stavu a funkcích daného prvku. V případě směrovače může agent zaznamenávat informace o přenosech dat, které prochází směrovačem, o dalších měřitelných veličinách a chování prvku za určitých podmínek. Dalším úkolem agenta je správa již zmíněné databáze MIB. Agent pomocí MIB sleduje a aktualizuje data. Data jsou uspořádána do struktury, jejíž části jsou datové objekty. Když si správce potřebuje určitou informaci, jednoduše zašle agentovi požadavek. Důležité je poznamenat, že správce může požadovat i více datových objektů v jednom požadavku. Agent tedy přijme požadavek na určité informace. Prohledá databázi a nalezne hodnoty, které správce vyžaduje. Poté zformuluje odpověď a zašle ji zpět správci. Ten ji přijme, dešifruje a na výstupu zobrazí v textové nebo grafické podobě. Data je potom možné jednoduše analyzovat. Jednotlivé agenty spravuje aplikace, které se mohou z hlediska složitosti lišit. V jednodušším případě provádějí dotazy a administrátorovi poskytují jen základní informace. Naopak ve složitějším případě mohou provádět dohled nad topologií sítě, sledovat měřitelné veličiny a informovat o požadovaných událostech. V nejsložitějších případech lze provádět i plánování, které najde uplatnění při budoucích úpravách a optimalizacích sítí. [1]

1.2 Komunikace SNMP

Vlastní komunikace probíhá prostřednictvím vyměňování SNMP požadavků a to mezi správcem a agenty. Tyto informace jsou přenášeny v podobě datové části paketu IP-UDP. Požadavky na informace jsou zasílány správcem pomocí datagramového protokolu UDP na portu 161. Tento požadavek je složen ze dvou částí, jimiž jsou hlavička a blok objektů. Hlavička nese informaci o verzi použitého protokolu SNMP, velikosti a heslo. Blok objektů obsahuje jeden nebo více požadovaných objektů a paket odpovědi. Vlastní komunikace mezi správcem a agentem probíhá pomocí sady operací, pomocí kterých je možné vytvářet požadavky a odesílat informace. Tyto operace jsou následující: [1]

GetRequest – tato operace je inicializována správcem a slouží k získání hodnot objektů od agenta. Agent, který tuto zprávu obdrží, zkontroluje, jestli není chybová, vyhledá v databázi požadované hodnoty a jako odpověď správci zašle zprávu s názvem GetResponse.

GetResponse – tato zpráva je zaslána agentem správci. Obsahuje vyhledaná data z databáze MIB. Když agentovi dorazí požadavek GetRequest obsahující chybu, je zpráva GetResponse nulová.

GetNextRequest – Tato zpráva je vysílána správcem za účelem získání objektů od agenta. Požadované hodnoty jsou uloženy tabulce. Hodnoty jsou zasílány postupně právě pomocí této zprávy. Agent na každou zpávu GetNextRequest odpoví zprávou GetResponse.

SetRequest – Úkolem zprávy SetRequest je upravování hodnot objektů na agentovi. Agent, který takovouto zprávu přijme provede změnu a odpoví opět zprávou GetResponse.

Depeše – zprávy tohoto typu jsou typické pro agenta. Ten odesílá tuto zprávu, aby upozornil na jistou událost. Od ostatních zpráv se tento typ značně liší, a to především tím, že ji generuje agent bez výzvy od manažera. Je zasílán po portu 160 datagramovou službou UDP. Další zajímavostí je, že každá zpráva obsahuje hlavičku obsahující adresu agenta, časové razítko a dále také obecné depeše. Existuje celkem 7 druhů těchto depeší:

ColdStart – nová inicializace agenta, konfigurace agenta,
WarmStart – nová inicializace agenta bez rekonfigurace,
LinkDown – chyba v připojení,
LinkUp – navázání spojení,
Authentication Failure – neúspěšná autentizace,
EGPNeighborLoss – selhání EGP agenta,
EnterpriseSpecific – upozornění správce na událost, která je doplněna dalším komentářem.

1.3 Objekty a MIB

Mezi objekty SNMP existují jisté operace, o nichž každý agent v síti má uložené informace. Jedná se o definici objektu a informaci o stavu. Existují případy, že dané zařízení podporují více než jednu MIB databázi. Ostatní databáze tedy mohou obsahovat nejen informace o agentovi samotném, ale navíc informace agentem shromážděné. Můžeme si to vysvětlit na ovladatelném přepínači. Ten může obsahovat data jako modelové a typové označení, verzi firmwaru, nebo informace o přenesených datech, chybně přenesených datech atpd. Správce si vyžádá informace a agent může jako odpověď zaslat buď požadované

informace, nebo ve zprávě zmínit chyby. Data jsou na straně správce vyobrazena a následně má správce možnost změnit konfiguraci v našem případě na řízeném přepínači. [1]

Object Syntax Notation – Jak už z anglického názvu vyplívá, úkolem zápisu ASN.1 je textově popisovat databázové objekty. ASN definuje jasná standardní pravidla pro kompatibilní MIB. Jsou definovány i znaky, jimiž jsou objekty popsány. Objekty SNMP je možné vyjádřit jako identifikátor OID s připojenou adresou „0". Objekt tedy může být vyjádřen například jako 1.2.6.3.2.1.1.1.0. Pro samostatné objekty je na konci vždy použita nula, aby byly odlišeny od druhého typu objektů, jimiž jsou sloupcové objekty, které na rozdíl od samostatných objektů nesou více hodnot. Kromě samostatných a sloupcových objektů existují ještě tabulky. Ty se používají v případech, kdy pro podobný znak existuje více datových položek. Příkladem může být směrovací tabulka.

Šifrování – Objekty databáze MIB jsou, jak již bylo zmíněno, pro přenos v síti speciálně formátovány. Pravidla, která popisují toto formátování, se nazývají základní pravidla šifrování BER (*Basic Encoding Rules*). BER vyjadřuje objekty v binární soustavě a používá se pro typy, hodnoty a ID objektů. Standardní zašifrovaná hodnota BER se skládá z pole o velikosti 1 bajt, proměnné, definice délky a datového pole.

Identifikátory – Databáze MIB je rozdělena na větve, které jsou identifikovány číselným parametrem. Proto i každý objekt má svoje číselné označení, jedinečnou adresu, které říkáme identifikátor objektu (Object Identifier - OID).

1.3.1 Skupiny MIB

System – skupina obsahující informace o hardwaru a softwaru,
Interfaces – vlastnosti rozhraní,
At – informace pro překlad adres,
Ip – informace potřebné pro použití protokolu IP,
Icmp – skupina obsahující informace potřebné pro použití protokolu ICMP,
Tcp – informace související s využitím protokolu TCP,
Udp – skupina nesoucí informace pro použití UDP protokolu,
Egp – skupina nesoucí informace o stavu při použití protokolu EGP,
Transmission – skupina poskytující informace o přenosech,
Snmp - skupina, jejíž úkolem je zpracování objektů za účelem shromažďování Informací.

1.4 RMON (Remote Network Monitoring)

Protokol SNMP využívá principu klient/server. Správce představuje server a objekty v síti jsou klienti. V některých případech může toto schéma zapříčinit nadmírné zatížení. Jedná se o případ, kdy jsou agenti neustále dotazováni. Právě tento problém poskytl motivaci pro vývoj metody pro sledování síťových prvků a to RMON (*Remote Network Monitoring*). U

této techniky jsou role klient/server prohozeny. Jako server zde účinkuje agent a roli klienta zde hraje správce. Již není nutné, aby správce shromažďoval veškerá data, agenti provádí kompletní zpracování. Agent odesílá správci depeše a informuje jej tak, že dojde k nějaké události.

Objekty metody monitoringu se od protokolu SNMP liší především vyšší mírou inteligence. [1]

Statistics – podobně jako *interfaces* u SNMP, tedy informace o rozhraních,
History – určení metody kolekce informací,
Alarm – signalizace daných událostí,
Hosts – monitoring MAC adres hostitelů v síti,
hosttopN – monitoring statistických informací u řízených hostitelů,
Matrix – monitoring informací o přenosech mezi hostitely,
Filter – skupina sloužící k výběru paketů,
Capture – skupina která řídí zachycování paketů,
Event – skupina ovládající generování událostí.

1.5 Ostatní protokoly

Kromě nejznámějších protokolů pro správu a monitoring SNMP a RMON existuje ještě řada obdob jako jsou DMTF, http nebo WEBM, jejichž znalost není na škodu.

DMTF (*Desktop Management Task Force*) je skupina zrozená v roce 1992, skládající se ze dvou částí DMI (*Desktop Management Interface*) a MIF (*Management Information Format*). Tato skupina představuje proces správy hardwaru a softwaru nezávislý na platformě. Část DMI funguje jako spojení mezi hardwarem, softwarem a programy pro správu. Druhá část, MIF představuje jazyk.

Je důležité podotknout, že i protokol HTTP je částečně využíván pro správu, a to určitými programy. [1]

2 Rozhraní API

Pod zkratkou API (*Application Programming Interface*) si můžeme představit rozhraní využívané k programování aplikací. Pojem API je často využíván v oboru softwarového inženýrství a jednoduše řečeno, jedná se o soubor funkcí a procedur určité knihovny nebo nějakého programu, které programátor může volně využít. API definuje způsoby volání funkcí ze zdrojového kódu aplikace a výsledkem tedy je, že je tento zdrojový kód možné zkompilovat. S pojmem API souvisí i nízkoúrovňové rozhranní mezi aplikacemi a systémem nebo knihovnami zvané ABI (*Application Binary Interface*). To je vytvářené při kompilaci a na rozdíl od API, je jeho úkolem umožnit zkompilovanému kódu funkci bez jakýchkoliv změn na systému s kompatibilním ABI. API definuje rozhraní pro interakci s určitými funkcemi, které jsou využívané systémem, existuje ve čtyřech variantách:

Obecné API: API je popsáno jedním konkrétním souborem obsaženým v knihovnách daného programovacího jazyka (Java API),
Konkrétní API: určeno pro řešení jednoho problému (Google Maps),
Jazykově závislé API: toto API je dostupné pouze v určitém jazyce,
Jazykově nezávislé API: na rozdíl od jazykově závislého lze toto API volat z více programovacích jazyků. To je vhodné pro službu, která na daný systém není vázaná.

Jako příklad můžou být webové stránky, které návštěvníkům umožní vyhledání místních divadel. Aplikace si přebere vrstvu z GoogleMaps, jelikož API těchto map to umožňuje. Rozhraní API lze využít i k odkazu na rozhraní, funkci nebo více API.

Dalším příkladem mohou být například známá grafická API OpenGL nebo DirectX. Ty může programátor při vývoji softwaru volně využít a nemusí tedy trávit čas nad zvláštním programováním funkcí obsažených v daném API. [2]

2.1 API v objektově orientovaných jazycích

Úkolem API v objektově orientovaném jazyce je důkladně popsat soubor, třídy a chování, což je několik pravidel, které popisují, jak se bude daný objekt vycházející z dané třídy chovat. API zde představuje soubor všech možností veřejných tříd, běžně nazývaný jako třída rozhraní. API v tomto případě lze pochopit jako soubor objektů, které lze pomocí metod odvodit z definic tříd. Metoda zde představuje technický detail popisující implementaci chování.

Jako příklad můžeme uvést objektově orientovaný jazyk Java a jeho JavaAPI. V tomto API je obsaženo rozhraní *Serializable*, které vyžaduje třída implementující chování v serializovaném módu. Není nutné mít v tomto případě veřejné metody, ale je nutné, aby třída měla povolení zastoupení, které je možno kdykoliv uložit. To umožňují třídy *containing*. Ty jsou jednoduché, datové a navíc nemají vazbu na externí zdroj.

API v objektově orientovaných jazycích představuje soubor pravidel chování, popřípadě soubor třídy metod. API je distribuováno jako knihovna. U jazyka Java je to knihovna JDK (*Java Developement Kit*), kterou programátoři používají při tvorbě softwaru. [2]

3 Platforma MIKROTIK

Když se řekne Mikrotik, snad každému technikovi znalému v oboru bezdrátových počítačových sítích ihned vysvitne myšlenka na malou velmi šikovnou krabičku zvanou RouterBoard, které jsou v hojném počtu nasazovány do meších i větších Wi-Fi sítí o stovkách až tisících uživatelů. Z hlediska softwarové stránky je RouterBoard založen na Linuxu a proto je tedy jeho konfigurace velice snadná.

Za posledních několik let tato platforma silně nabírala na popularitě a v současnosti je na vrcholu stupnice poměru cena-výkon. Ačkoliv je Mikrotik již zavedený název a mezi technikami představuje desku tištěného spoje osazenou bezdrátovými mini-PCI kartami je toto označení nesprávné. Mikrotik je název firmy vyvíjející zařízení RouterBoard a programovou část RouterOS, nikoliv samotný hardware. Vznik této firmy se datuje do roku 1995 a její sídlo je v Lotyšsku. [4]

3.1 Pojmy

3.1.1 RouterBoard

Pod pojmem RouterBoard (RB) si tedy můžeme představit desku, na níž se nacházejí nejrůznější elektronické součástky. Disponuje jedním nebo více mini-PCI sloty nejčastěji pro osazení bezdrátovými kartami, porty pro připojení síťového kabelu RJ-45 a portem pro připojení sériové konzole RS-232. Mozkem tohoto zařízení je procesor, operační paměť a paměť flash s přeinstalovanou licencí softwaru RouterOS. Rozhraní RS-232 je ač by se mohlo zdát na první pohled nepotřebné, avšak opak je pravdou. Toto rozhraní uživatel velmi ocení v případě, že při konfiguraci udělá chybu, do zařízení se není možné přes standardní RJ-45 přihlásit a volba resetu do základního továrního nastavení by přinesla značné komplikace a to v podobě opětovné konfigurace zařízení. [4]

3.1.2 RouterOS

Router Operating System je program nahraný v RB, jehož úkolem je dle předem definovaných konfigurací a pravidel řídit větší nebo menší část sítě. Operační systém disponuje nepřeberným počtem funkcí, které zde kvůli jejich množství nebudou moci být všechny představeny. RouterOS je podobně jako tomu je například u řešení Microsoft Windows XP dostupný v několika verzích, které se liší funkčně a tudíž i cenově. Stručný přehled funkcí a omezení je vidět na tabulce č. 1.

Možnosti	L1 (zdarma)	L2 (Demo)	L3	L4	L5	L6
aktualizace	ne	ne	ROS v4.x	ROS v4.x	ROS v5.x	ROS v5.x
podpora	ne	ne	ne	15 dní	30dní	30dní
wifi AP	24h	ne	ne	ano	ano	ano
wifi klient	24h	ne	ano	ano	ano	ano
směrování	24h	ne	ano	ano	ano	ano

Tabulka 1: verze licencí systému RouterOS a stručný přehled funkcí

Předcházející tabulka je stručným porovnáním možností a funkcí jednotlivých licencí dostupných v přeinstalované formě na deskách RouterBoard nebo na samostatných modulech flash. Základní licence L1 je pouze testovací verzí která po 24 hodinách automaticky přestane fungovat. Po tuto dobu však lze využít veškeré funkce a proto se hodí pro uživatele jako ukázka veškerých funkcí a možností. L2 je verze, která od své instalace vybízí uživatele zadat registrační kód. Funkce jsou bez registrace velmi omezené. Není možné využít router jako bezdrátový prvek nebo aplikovat dynamické směrování OSPF(*Open Shortest Path Firs*t), RIP (*Routing Information Protocol*). Tuto variantu lze doporučit pouze jako stručnou ukázku, po registraci ji lze plně využívat. Licence L3 nejsou dostupné pro individuální koupi. Minimální odběr je stanoven na 100 kusů. Na RouterBoardech ji však v přeinstalované formě lze zakoupit. U této verze je možné zařízení využít v bezdrátové síti, ale pouze v módu klient. Jako přístupový bod AP nelze zařízení s touto licencí nastavit. Nevýhodou je také fakt, že uživatel nemůže využít technické podpory. Na vyšší úrovni je verze L4, která je již plně podporována a navíc lze zařízení nakonfigurovat do všech známých bezdrátových módů, včetně přístupového bodu AP. U této verze je možné mít až 20 přihlášených administrátorů, kterým lze přidělovat určitá práva, podobně jako tomu je u nastavení práv uživatelů přihlašujících se na FTP server. U licence L3 je tento počet poloviční. Cena této licence se pohybuje kolem 850,- Kč. Druhá nejvyšší licence je L5, která má 30 denní možnost základní podpory prostřednictvím e-mailu. Počet současně přihlášených správců je omezen na 50. Další omezení je třeba v počtu PPPoE (*Point-to-Point Protocol*) záznamů. Toto však může být zásadní pouze pro velmi náročného uživatele. Proto se v naprosté většině v praxi setkáváme s licencí L4, která je svými vlastnostmi dostačující. Nejvyšší licence je označována jako L6. Tato verze není krom omezení 30 denní podpory nijak omezená. Jedná se o nejvýkonnější, avšak i nejdražší licenci, kterou lze zakoupit samostatně ale i v přeinstalované formě na těch nejvýkonnějších deskách RouterBoard. Cena této licence je kolem 3 tisíc.[4] [5]

3.2 Modely a typy

Kromě varianty předinstalovaného operačního systému RouterOS se jednotlivé modely RouterBoard liší především výkonem osazeného procesoru, kapacitou flash a operační paměti a možnostmi rozšíření. K některým modelům se dá v případě potřeby připojit takzvaný DoughterBoard co je deska, která na sobě nese určitá rozšíření například sloty pro další mini-PCI karty nebo porty RJ-45. Každá řada výrobků RouterBoard se liší také označením například RB532, kde první číselná hodnota označuje modelovou řadu, která je pro výběr správného zařízení nejdůležitější.

RB1xy – toto označení nesou nejnižší modely, které jsou vhodné pro nasazení spíše na strany klientů než pro stavbu komplikovanějších struktur.

RB3xy – modely s tímto typovým označením jsou nástupci řady 1. Disponují o něco výkonnějšími čipy, a proto se hodí i pro menší páteřní spoje

RB4xy – tato modelová řada je nástupcem řad 1 a 3, které v současnosti již nevyrábí. Uplatnění nachází jak v domácím prostředí, tak ve vytížených sítích.

RB6xy – tato řada je druhou nejvýkonnější, kterou firma Mikrotik představila. Je určena především pro stavbu bezdrátových páteřních spojů.

RB10xy – Nejdražší, však nejvýkonnější řada nasazována především pro vysokorychlostní a velkokapacitní spoje ať už metalické nebo bezdrátové.

Je důležité si uvědomit, že Mikrotik nemá jedno možné využití, které je nutno s jistými rezervami dodržovat. Jedná se o platformu otevřenou k již známým ale i novým, vlastním konfiguracím, které si uživatel může sám vymyslet a pomocí tohoto zařízení zrealizovat. [4]

3.3 Výhody vs. nevýhody

Platforma Mikrotik má samozřejmě svoje světlé i tmavé stránky. Mezi výhody můžeme zahrnout: poměr cena/výkon, účinný firewall, několik možností konfigurace, rychlost, stabilita, proxy, dynamické směrování, podpora mnoha standardů 802.11, podpora, dostupná dokumentace. Z nevýhod je nutné zmínit fakt, že Mikrotik neuvolnil zdrojové kódy, i když se jedná o systém založený na Linuxu. Nevýhodou je i nemožnost aplikace vlastních skriptů v systému RouterOS. Konfigurační utilitu Winbox lze navíc spouštět pouze z operačního systému MS Windows. [4]

3.4 Konfigurace a správa

Platforma je velmi otevřená i z hlediska možností připojení a správy. Disponuje 4 možnostmi správy a 3 možnostmi připojení. Nejjednodušší připojení je pomocí sériové konzole, jehož výhoda je, že uděláme-li při konfiguraci chybu, RouterBoard nás z nastavení nevyhodí. Další možností připojení je pomocí klasického Ethernetu kabelem RJ-45 nebo bezdrátově pomocí invertované Wi-Fi karty. Ke správě můžeme využít telnet, SSH, Winbox nebo Webbox a jako komunikační kanály lze použít IP/TCP, linkovou vrstvu nebo již zmíněný sériový kabel. [4]

Konfigurace přes SSH a telnet je stejná. Rozdíl je v zabezpečení jak již vyplývá z názvu SSH (*Secured Shell)*. Od verze Router OS 3.0 je u správy pomocí příkazů pomoc ve formě intuitivního doplňování neúplně zadaných příkazů, či zobrazení dostupných příkazů.

Winbox je grafická utilita s velkým množstvím funkcí určená pro správce, kteří se lépe orientují spíše v grafickém rozhranní než v příkazové řádce. Chceme-li využít vrstvu IP/TCP stačí mít na RouterBoardu a na PC správně nakonfigurovanou IP adresu. Potom už stačí jen jednoduše pomocí Winboxu nebo SSH zadat IP adresu, jméno a heslo. Připojení pomocí linkové vrstvy přijde vhod uživateli v případě, že na RouterBoardu není nakonfigurována IP adresa a nemá po ruce kabel RS-232, jednoduše řečeno RouterBoard jsme pouze připojili na napájení a k síti LAN. Na PC stačí spustit WinBox provést hledání a program sám nalezne dostupné RouterBoardy pomocí jejich MAC adres. Zde je nutno podotknout že pro tuto možnost je nutné, aby na PC byl operačním systémem Windows, a aby IP adresa PC byla ze stejného segmentu sítě. Výhodou je, že když provedeme rozsáhlou konfiguraci, můžeme si ji vyexportovat a zálohovat pro případ selhání zařízení, nebo ji nahrát do jiného zařízení a mít tak dva identicky nastavené směrovače a mít tak pojistku v případě havárie. [4]

Závěrem je třeba dodat že operační systém Router OS není jediným řešením, které je možné na RouterBoardu používat. Je možné využít, některé distribuce Linuxu, které by byly pro mnohé správce sítí pohodlnější. Bohužel však zařízení disponuje pouze omezenou pamětí, kterou nejde nijak rozšířit (až na některé starší modely, které obsahovaly slot na CF kartu např. RB 532A). Na druhou stranu i RouterOS je možné využít i na jiných zařízeních než na RouterBoardech. Jelikož je dostupný i pro architekturu x86 je možné ho nainstalovat na libovolný počítač.

Obrázek 1: bezdrátový směrovač Mikrotik RB433

4 Open WRT a DD-WRT

Pod pojmem OpenWRT si můžeme představit, jak už český překlad napovídá, volně šiřitelnou aplikaci či software. A skutečně, jedná se o linuxovou distribuci firmwaru, která se používá převážně na síťových zařízeních, například na přístupových bodech, směrovačích nebo rezidentních branách. Tento software je tedy postaven na linuxovém jádře a skládá se ze stále rozrůstající se sbírky různých softwarových balíků. Pro správu celého systému, jednoduchou instalaci a odinstalaci jednotlivých balíků se využívá systém opkg. Jednotlivé funkce lze konfigurovat přes příkazovou řádku nebo BASH, pro pohodlnější nastavování lze využít i grafické webové prostředí jako je LuCI nebo X-Wrt.

V současnosti je kvalita celého projektu již na vysoké úrovni. Nabízí mnoho funkcí a možností, jejichž limitem jsou omezené schopnosti hardwaru. Výhodou celého projektu je mnoho možností podpory. Na webových stránkách lze nalézt knihovnu Wiki, forum nebo IRC kanály pro dodatečnou technickou podporu. [3]

4.1 Vznik OpenWRT

Celý projekt vznikl, jak už tomu v historii několikrát bylo, v důsledku neúmyslné chyby společnosti Linksys, která je dceřinou společností společnosti Cisco. Firmware byl postaven na základě modifikovaného kódu GPL (Genreal Public Licence), což je licence pro veřejně dostupný kód a tak byli vývojáři nuceni kompletní zdrojový kód a modifikace uveřejnit. Zprvu bylo podporováno pouze zařízení Linksys WRT54G, ale brzy byl seznam podporovaných zařízení a jejich chipsetů velice rozšířen. Vývoj OpenWRT byl podporován jednoduchostí jednotlivých modifikací a kvůli použití licence GPL byli vývojáři průběžná vylepšení a modifikace nuceni zveřejnit. Před uveřejněním verze OpenWRT 8.09, která byla založena na jádru 2.6.25 a modulu jádra b43 bylo bezdrátové rozhraní některých směrovačů založených na procesoru Broadcom dostupné pouze s aplikací proprietárních wl.0 modulů. Tyto moduly byly dostupné pouze ve starší verzi jádra 2.4. [3]

4.2 Hlavní znaky a funkce

Firmware OpenWRT poskytuje prakticky všechny funkce jako běžný komplexní firmware v zařízeních jako jsou bezdrátové přístupové body. Jedná se například o dynamické přidělování síťových adres DHCP, šifrovací algoritmus WEP nebo pokročilejší metodu zabezpečení WPA2 (*Wifi Protected Access*). Navíc však poskytuje mnoho funkcí, které jsou často chybně provedeny, nebo dokonce chybí. Některé z těchto funkcí umožňují transformovat na první pohled obyčejné zařízení na velmi sofistikovaný síťový prvek. Je však nutno podotknout, že již samotné Linuxové jádro poskytuje velmi důmyslné metody určené k správě sítí, jakými jsou například:

- Metody konfigurace sítě zahrnující sofistikované metody nastavování virtuálních sítí VLAN.

- Metody filtrování – firewall, tzv. port forwarding (nastavení směrování komunikace skrz směrovač na určitém portu), NAT- překlad síťových adres.
- Metody pro podporu QoS (Quality of Service), jejichž úkolem je optimalizovat síť pro VoIP, hraní her, sledování streamového videa.
- Omezování přenosových rychlostí jednotlivých klientských stanic (využití například u poskytovatelů internetu ISP), Load Balancing- řízení zátěže.
- Monitoring parametrů sítě, statistické vyhodnocování hodnot.
- Metoda dynamického přidělování adres serverů DNS (*Domain Name Server*), podpora UPnP (*Univeresal Plug and Play*) – sada síťových protokolů pro jednoduché připojení komponent.
- Konfigurace zařízení jako bezdrátový opakovač, přístupový bod, most.
- Metody pro provádění oprav a údržby systému, aktualizace systému.

Velmi výhodnou vlastností firmwaru OpenWRT je jeho plně zapisovatelný souborový systém. Umožňuje jednoduchou správu balíčků a právě tím je velmi univerzální a adaptovatelný na různé požadavky. OpenWRT tedy na rozdíl od jiných typů firmwarů založených na Linuxovém jádru umožňuje uživatelům instalovat nový software. Existují totiž firmwary s Linuxovým jádrem, avšak založené na systému souborů, který poskytuje pouze práva pro čtení (například SquashFS). Ten nabízí efektivní metody komprese, ale neexistuje u něj způsob, jak pozměnit nainstalovaný software, aniž bychom byli nuceni systém kompletně přestavět a přehrát celý obraz firmwaru. Firmware OpenWRT dosahuje velmi slušné úrovně komprese a jeho souborový systém nese označení JFFS2. [3]

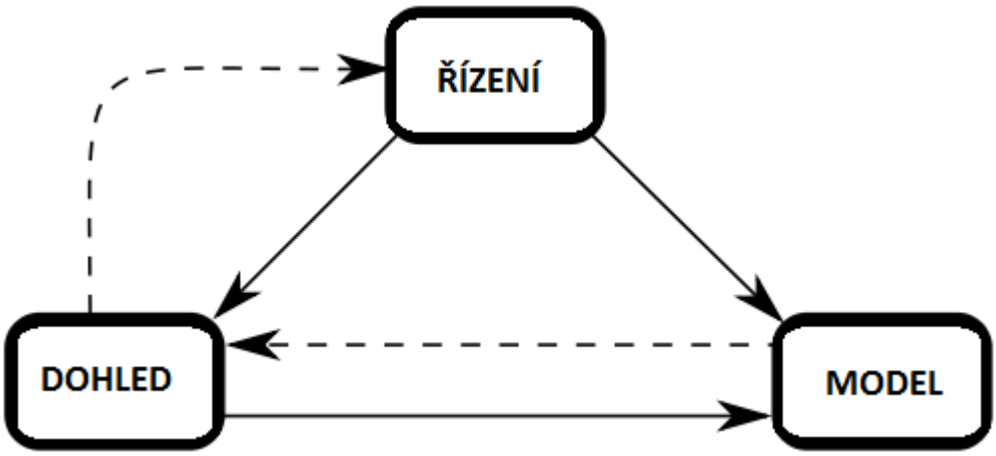
4.2.1 Webové rozhraní

Možnosti webového rozhraní první verze firmwaru byli velmi omezené. Teprve až od verze 8.09 lze využít propracovanější webové rozhraní založené na tzv. LuCi, architektuře model-dohled-řízení MVC (Model-View-Controller). Alternativou je také tzv. X-Wrt webové rozhraní.

Obrázek 2: uživatelské webové rozhraní OpenWRT

MVC je softwarová architektura, v současnosti považovaná za vzor v oblasti softwarového inženýrství. Odděluje aplikační logiku od uživatelského rozhraní a umožňuje nezávislý vývoj a testování. Jak již z názvu vyplývá, je architektura popsána třemi pojmy: model, dohled, řízení. Úkolem modelu je řídit chování a data aplikační domény. Odpovídá na požadavky na informace o stavu (dotazy z části dohledu) a reaguje na pokyny ke změně stavu (pokyny z části řízení). Jedná se o systém řízený událostmi, model upozorní část dohledu, že došlo ke změně a ten potom může zareagovat. Úkolem dohledu je vykreslit model do podoby vhodné pro interakci. Může existovat i více než jeden modul dohledu, přičemž každý z nich má jiný úkol. Jednotka řízení přijímá vstupy od uživatele a iniciuje reakci tím, že dotazuje objekty modelu.

S architekturou MVC se v praxi můžeme setkat například u webových aplikací, kde část dohledu představuje HTML popřípadě XHTML kód generovaný samotnou aplikací. Řízení dostává informace ze vstupu a rozhoduje se, jak s nimi naloží. Předává informace objektům domény (modelu), které obsahují pravidla pro řešení zvláštních úloh. [3]



Obrázek 3: schéma architektury MVC

4.3 DD-WRT

Podobně jako tomu bylo u OpenWRT i DD-WRT je firmware založený na Linuxovém jádře s volně šiřitelnou licencí. Je určený k aplikaci na mnoho typů bezdrátových routerů založených především na chipsetech od firmy Broadcom nebo Atheros. Tento volně šiřitelný firmware uživatelé často nasazují namísto původního-originálního firmwaru dodávaného se zařízením a to z několika důvodů. Jedním je například fakt, že firmware DD-WRT obsahuje další dodatečné funkce, které se u originálních firmwarů v naprosté většině neobjevují.

Od první verze až do verze DD-WRT v. 22 byl firmware založen na firmwaru zvaném Alchemy, o jehož vývoj se postarala firma Sveasoft Inc. Ten je zase založen na původních firmwarech firmy Linksys a na řadě volně dostupných projektů.

Současná verze firmwaru DD-WRT označovaná také jako *DD-WRT v24* představuje zcela nový projekt. Firmware nabízí mnoho funkcí a možností konfigurace navíc k běžným funkcím, které můžeme nalézt v průměrném routeru. [6]

Obrázek 4: bezdrátový směrovač značky Linksys určený pro OpenWRT nebo DD-WRT

Obrázek 5: ukázka webového uživatelského rohzraní DD-WRT

5 Rozbor praktické části

Cílem praktické části tohoto semestrálního projektu bylo pro daný přístupový bod či platformu pro bezdrátovou komunikaci realizovat aplikaci, která umožňuje pravidelně změřit objem dat přenesených při komunikaci s konkrétním bezdrátovým klientským zařízením v nastavitelných časových intervalech. Jelikož nebylo konkrétně řečeno jakým způsobem je třeba uvedená data získat a jaké by měli být viditelné výsledky či formát hodnot objemů přenesených dat, byl celý úkol pojat komplexně a to tak, že by se ve výsledku jednalo o jednoduchou administrátorskou aplikaci, která by uživateli umožňovala monitorovat dané veličiny. Jako prostředník pro sběr informací by měl posloužit protokol SNMP, který již byl popsán v teoretické části. Aby celá aplikace v konečné podobě působila na uživatele sofistikovaně, bude doplněna o několik vlastností, které by správce sítě mohl ocenit. Byly stanoveny následující požadavky:

- monitorování důležitých stavových hodnot zařízení
- monitorování přenesených dat klientů (download/upload)
- generování grafů z naměřených dat
- snadné uživatelské rozhraní pro definování a editaci (nové zařízení, vložení klienta, zadání požadavku na měření hodnot v určitém intervalu)
- pro zjednodušení vkládání klientů do jisté míry automatizovat nastavování zařízení

5.1 Rozvaha nad řešením projektu

Klíčovým rozhodnutím na začátku vývoje byl výběr platformy. S ohledem na osobní zkušenost se známými programovacími jazyky připadaly v úvahu následující platformy:

- Java (*Sun Microsystems*)
- C/C++
- PHP (*Hypertext Preprocessor*)

Všechny uvedené možnosti mají spoustu výhod i nevýhod. Pro Javu hovoří komplexnost celé platformy a relativní rychlost a efektivita vývoje. Její problém tkví v tom, že Java nepodporuje práci s protokolem SMNP, což by vyžadovalo dovyvinutí této podpory. Java byla tedy zamítnuta s ohledem na čas, který by byl pro realizaci projektu zapotřebí.

C/C++ má výhodu v tom, že běží jako nativní aplikace na daném počítači. To znamená, že výsledný program je kompilován přímo pro daný konkrétní operační systém. Na rozdíl ostatní řešení (Java, PHP) jsou interpretována, nebo běží ve virtuálním stroji. Zjednodušeně můžeme říci, že jsou multiplatformní. Výhodou řešení úkolu v programovacím jazyce C/C++ by tedy byla rychlost samotné aplikace. Na druhou stranu je nutno podotknout, že časová náročnost vývoje samotného projektu by byla velmi vysoká. Krom toho by bylo

nutné dostudovat potřebné knihovny, jelikož jazyk C/C++ neumí použít protokol SNMP. Na základě těchto nevýhod byla tato možnost rovněž zavrhnuta.

Třetí, poslední možnou variantou je skriptovací programovací jazyk, určený především pro programování dynamických internetových stránek, PHP. Po zvážení okolností, podmínek projektu a všech vlastností této platformy vyšlo PHP jako vítěz. Jednoznačnou výhodou je zde podpora a spolupráce s protokolem SNMP. Není nutná instalace a studování speciálních knihoven, takže není problém začít tento protokol rovnou využívat. Další výhodou je rychlost, se kterou se PHP skripty dají tvořit a všude dostupná podpora tohoto jazyka. Po důkladném zvážení všech aspektů tyto argumenty převážily a varianta PHP pro implementování celého projektu vyšla jako nejlepší.

5.2 Platforma pro běh a vývoj programu

Kromě výběru programovacího jazyka bylo také nutno rozhodnout, na jakém operačním systému bude projekt vyvíjen. V úvahu připadaly pouze dvě možnosti a to GNU/Linux a Microsoft Windows. Pro MS Windows hovoří všeobecná rozšířenost a fakt, že je obecně společností vnímán jako jakási standardní volba pro pracovní stanici. Bohužel se jedná o uzavřený operační systém, který je už z principu vyvíjen pomaleji než volně šiřitelné (*open source*) alternativy. Pro současného největšího konkurenta operační systému MS Windows, tedy GNU/Linux, hovoří cena a otevřenost celé platformy, která zaručuje bezpečnostní aktualizace, větší stabilitu a podporu. Z hlediska vývoje by nějaké větší změny mezi uvedenými platformami nebyly patrné. Z hlediska běhu aplikace by však mohli být oceněny právě uvedené vlastnosti GNU/Linux. Nakonec po zvážení všech aspektů, okolností a požadavků byl vybrán GNU/Linux jako platforma pro vývoj i pro běh celé aplikace. Linux není samozřejmě mezi běžnými uživateli tak znám jako operační systém Microsoft Windows, ale vyvíjený systém bude serverová aplikace, u které je tržní podíl na stanicích nepodstatný.

5.3 Architektura projektu

Celý projekt lze rozdělit do tří hlavních částí, které budou dále podrobněji popsány. Jedná se o:

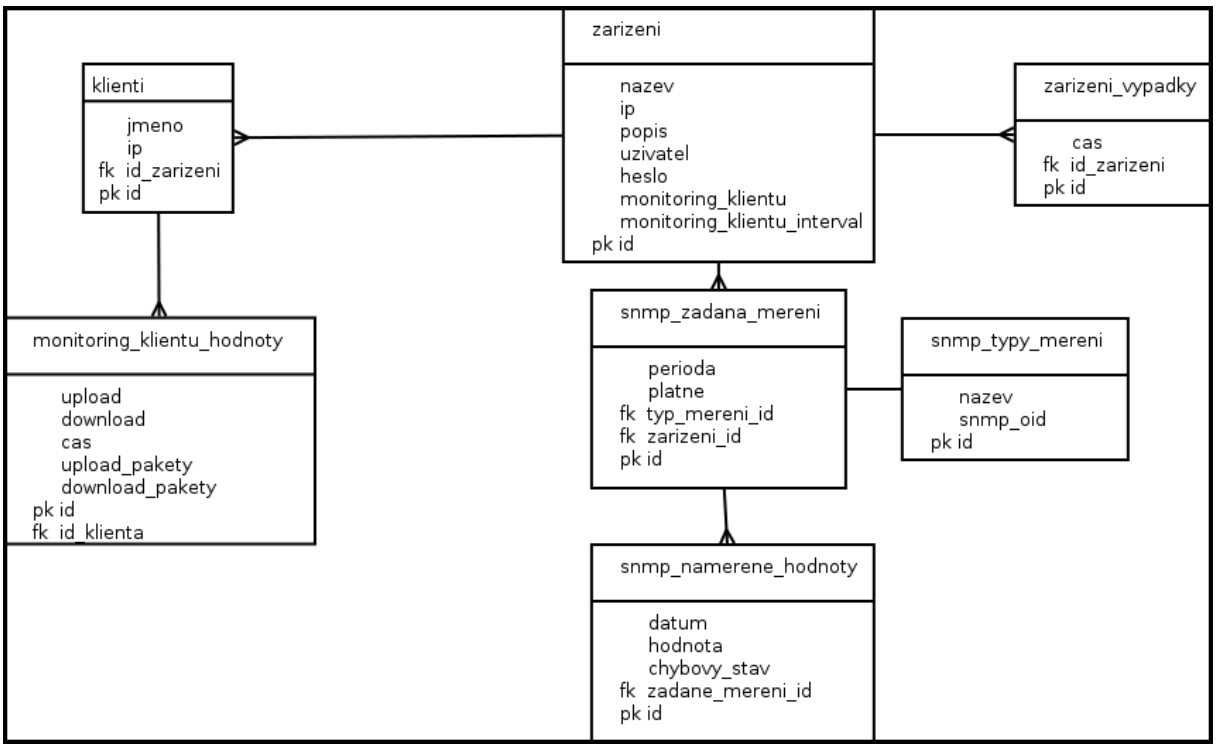
- databáze
- webové rozhraní (uživatelské)
- tzv. backend

Důležité při návrhu projektu bylo vzít v potaz fakt, že aplikace bude v budoucnu doplňována o funkce, či jakkoliv jinak vylepšována a upravována (aplikační rozhraní aplikace místo webového, nové měřící metody atd.). Proto je nutné oddělit vlastní data, měřící aplikaci a zobrazovací rozhraní. Výhodou aplikace je fakt, že celá aplikace se skrývá v jediném adresáři. Aplikaci nebude nutno nijak instalovat, a až na několik nastavení bude možno aplikaci jednoduše spouštět nakopírováním složky. Soubory budou standardně uloženy v adresáři **/var/www/netmon**.

5.3.1 Databázová část

Databáze projektu slouží jako kostra celého systému. Jsou zde uchovávány všechny naměřené hodnoty a nastavení celé aplikace. Databázová část projektu musí běžet na databázovém stroji (slang. enginu), což je síťový server, který řídí komunikaci s databázovými klienty, což jsou v tomto případě další části projektu. Navíc musí poskytovat data z databáze dle klientských požadavků. V tomto případě padla volba na databázový „engine" MySQL od firmy Oracle (původním vlastníkem byl Sun Microsystems). Toto řešení bylo vybráno zejména kvůli svobodné licenci, která umožňuje bezproblémové nasazení, a podpoře.

V následujícím diagramu je vidět struktura tabulek databáze. Diagram byl ručně vytvořen programem Dia, který je v Linuxu standardním nástrojem nejen pro tvorbu databázových diagramů.



Obrázek 6: diagram struktury databáze

Tabulka „zarizeni" obsahuje informace o zařízeních, na kterých bude aplikace provádět monitoring. Důležitými atributy jsou zde „uzivatel" a „heslo", které využívá Mikrotik API pro přihlašování na zařízení. Atribut „monitoring_klientu" udává jestli bude prováděno sledování přenesených dat klientů nebo ne. Atribut „monitoring_klientu_interval" udává periodu tohoto měření.

Tabulka „klienti" obsahuje informace o klientech, kteří mají být aplikací monitorováni a váže se na tabulku „zarizeni" atributem „id_zarizeni", který říká, na jakém zařízení je klient umístěn.

Další je tabulka „monitoring_klientu_hodnoty”, která slouží k periodickému ukládání naměřených hodnot přenesených dat klientů. Tabulka se váže na tabulku „klienti” atributem „id_klienta”, který udává, pro jakého klienta byla daná hodnota naměřena. Atribut „upload” udává počet přenesených bajtů ve vzestupném směru a podobně i atribut „download” udává počet bajtů přenesených v sestupném směru. Atribut „cas“ udává datum a čas (datový typ datetime) provedení měření. Dále atribut „upload_pakety“ udává počet paketů přenesených ve vzestupném a atribut „download_pakety“ počet paketů přenesených v sestupném směru. Z položek uložených v této tabulce jsou generovány grafy přenesených dat klientů.

Další tabulka je „snmp_typy_mereni“. Ta obsahuje názvy a SNMP OID měření, které je možno nadefinovat pro monitoring stavových hodnot zařízení. Pod pojmem OID (*Object Identifier*) si můžeme představit identifikátor, který je přiřazen individuálnímu objektu z databáze MIB.

Dále je na diagramu vidět tabulka „snmp_zadana_mereni“. Ta obsahuje definovaná měření stavových hodnot zařízení s využitím protokolu SNMP. Atribut „perioda“ udává periodu měření. Atribut „platne“ je výčtový typ, který udává jestli má být zadané měření prováděno nebo ne. Tabulka se váže na tabulku „snmp_typy_mereni“ atributem „typ_mereni_id“, který udává, jaký typ měření má být prováděn. Tabulka se dále váže na tabulku „zarizeni“ atributem „zarizeni_id“, který udává, pro jaké zařízení má být měření prováděno.

Tabulka „snmp_namerene_hodnoty“ obsahuje výstup z měření stavových hodnot zařízení s využitím protokolu SNMP. Atribut „datum“ obsahuje datum a čas, kdy byla hodnota naměřena. Atribut „hodnota“ obsahuje hodnotu, kterou vrátilo zařízení na měřený SNMP OID. Atribut „chybovy_stav“ udává, jestli během daného měření došlo k chybě či nikoliv. Tabulka se váže na tabulku „snmp_zadana_mereni“ atributem „zadane_mereni_id“, který udává, pro jaké měření byla hodnota naměřena.

Poslední je tabulka „zarizeni_vypadky“. Ta obsahuje informace o výpadku zařízení, který byl zjištěn při monitorování přenesených dat klientů. Atribut „cas“ obsahuje datum a čas, kdy byl výpadek zaregistrován. Tabulka se váže na tabulku „zarizeni“ atributem „id_zarizeni“, který udává, u jakého zařízení k výpadku došlo.

5.3.2 Webové rozhraní

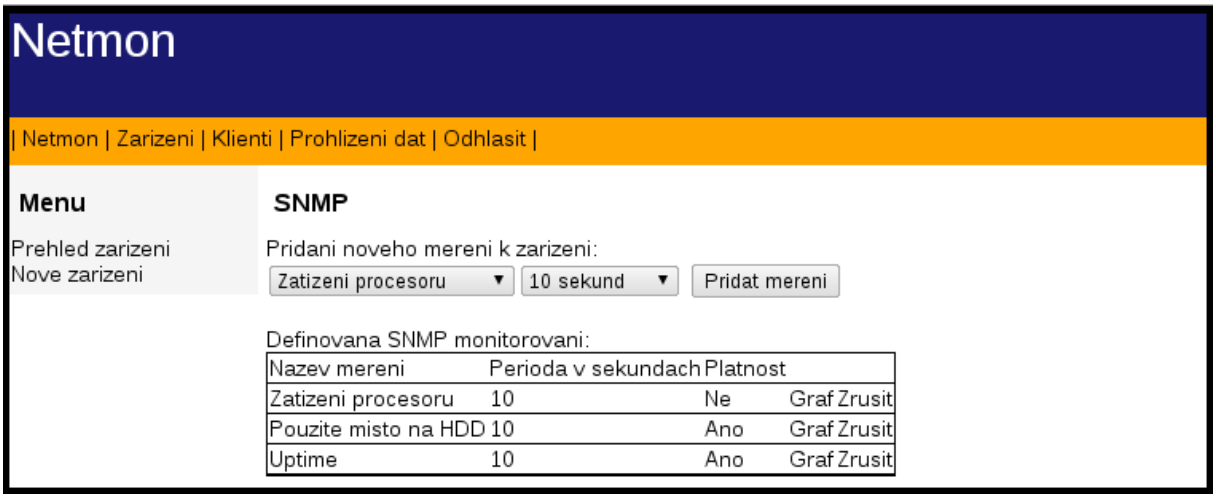
Ke správě aplikace a prohlížení naměřených hodnot slouží webové rozhraní. Je psáno rovněž v programovacím jazyku PHP, které je na podobná zadání velmi vhodné a to z hlediska rychlosti vývoje a nároků na programátora. Jelikož se jedná o velmi rozšířenou a všeobecně známou platformu, bude možné v budoucnosti aplikaci snadno dále rozšiřovat.

Webové rozhraní bylo vyvíjeno s důrazem na jednoduchost a přehlednost. Jelikož můžeme předpokládat, že uživatel aplikace bude využívat přístup k tomuto rozhraní přes síť Internet, nebyly při vývoji webové sekce použity žádné složité grafické prvky ani dynamické efekty, které by mohly zbytečně zatěžovat internetové připojení a zpomalovat práci se samotným rozhraním.

Uživatel může přes toto rozhraní nastavit celou aplikaci. Jedná se především o správu zařízení a klientů, zadávání a nastavení monitoringu zařízení a přenesených dat jednotlivých klientů, prohlížení naměřených hodnot a vizualizace grafů.

K samotnému generování grafů byla použita knihovna Jpgraph, která je velmi dobře dokumentována a hlavně volně dostupná. Svobodná licence této knihovny bez problému umožňuje nasazení ve vyvíjeném projektu. K běhu tohoto rozhraní je v aplikaci použit webový server Apache, který je standardní součástí všech hlavních GNU/Linuxových distribucí. Dalším důvodem, proč byl Apache vybrán z dlouhé řady volně dostupných webových serverů, bylo velké množství dokumentace a široká podpora. K webovému rozhraní je možno přistupovat zadáním adresy *http://localhost/netmon* do libovolného internetového prohlížeče, v případě, že obsluhujeme počítač, na kterém aplikace běží. V případě že obsluhujeme jiný počítač v síti, je třeba na místo *localhost* zadat IP adresu stroje, na kterém aplikace běží. Takto můžeme jednoduše zobrazit webové rozhraní na jiném počítači. Je tak vyřešena vzdálená obsluha, která usnadňuje konfiguraci.

Následující obrázek prezentuje vzhled webového rozhraní, pomocí kterého uživatel může monitorovat stanovené parametry a provádět konfiguraci.



Obrázek 7: ukázka webového rozhraní pro definici typu měření stavových hodnot

5.3.3 Backendová část

Jedná se o část aplikace, která běží na pozadí a uživatel s ní v podstatě vůbec nepřijde do styku. Zjednodušeně řečeno v backendu administrátor provádí veškeré úpravy a konfiguraci. Například u internetového obchodu backend slouží k vkládání zboží, jeho popis, nastavování ceny apod. Bez této součásti internetový obchod nemůže fungovat. Backend by měl nabízet množství funkcí a zároveň by měl být přehledný, tak aby se v něm dokázal zorientovat i laik. V tomto případě je hlavním úkolem backendové části zabezpečení vlastního monitorování a záznam naměřených hodnot do databáze.

Backend v tomto projektu je rovněž psán v jazyce PHP. Bylo zde využito faktu, že PHP samo o sobě obsahuje podporu pro práci s protokolem SNMP. Nebylo tedy třeba

používat pro práci s tímto protokolem podpůrné knihovny, ale stačilo použít standardní prostředky jazyka.

Spouštění backendu zajišťuje skript *backend.sh*, který v určené periodě vytvoří pro každé definované měření v databázi samostatnou instanci měřícího skriptu. Pro periody měření menší než 1 minuta je skript *backend.sh* spouštěn službou CRONTAB pouze jednou za minutu. Služba CRONTAB nebo zkráceně CRON je UNIXová utilita umožňující automatické spuštění úkolů, v našem případě skriptů v pravidelných a předem definovatelných intervalech. Toho uživatel může využít například při zálohování dat nebo právě při kontrole vzdálených zařízení. CRON je tabulka, která obsahuje určité údaje. Ty jsou spouštěny v definovatelném čase. Každý záznam má svůj řádek. Nevýhodou této služby je však fakt, že se hodnoty periodického spouštění nedají definovat pro sekundové periody. Proto je nutné pro tato měření skript spustit pouze jednou a měření o sekundových periodách vyřešit v rámci tohoto skriptu. Jakmile je vyslán požadavek na měření o periodě menší než jedna minuta například 10 sekund, skript provede měření, následně 10 sekund vyčkává a poté měření opakuje. V tomto případě v rámci jedné minuty tedy vždy provede 60/10 měření. Pro periody větší než jedna minuta je skript opětovně spouštěn vždy s danou periodou, a měření tedy proběhne vždy jednou v této periodě. K programování toho skriptu byl zvolen jazyk bash, protože jazyk PHP neumí asynchronní spouštění externích programů, které je nutné pro zajištění správné funkčnosti monitorovací činnosti. Monitorovací činnost lze rozdělit na dvě části:

- monitorování stavových veličin zařízení (monitoring_snmp.php)
- monitorování přenesených dat klientů (monitoring_klientu.php)

Níže uvedený úryvek zdrojového kódu pochází ze souboru *backend.sh*. zajišťuje načtení seznamu měření stavových hodnot zařízení a seznamu zařízení, pro které má být provedeno monitorování přenesených dat klientů. Skriptu je perioda měření předána jako parametr službou CRON. Pro každé definované měření je spuštěna samostatná instance měřícího skriptu (*monitoring_snmp.php* nebo *monitoring_klientu.php*) na pozadí (použitím modifikátoru „&“ za spuštěním měřícího skriptu). Jak již bylo řečeno pro periody větší než minuta, jsou skripty pouštěny opakovaně vždy po uplynutí periody a pro periody do jedné minuty je skript puštěn pouze jednou s tím, že mezi jednotlivým měřením je skript vždy uspán na dobu o délce stanovené periody.

```
# Nacteni vsech SNMP mereni pro zadanou periodu
SEZNAM_MERENI=`echo -n "select snmp_zadana_mereni.id as id from
zarizeni, snmp_zadana_mereni, snmp_typy_mereni where
snmp_zadana_mereni.typ_mereni_id = snmp_typy_mereni.id and
snmp_zadana_mereni.zarizeni_id = zarizeni.id and
snmp_zadana_mereni.platne = 'Y' and snmp_zadana_mereni.perioda =
\"$1\";" | mysql -B -h "$MYSQL_HOST" -u "$MYSQL_USER" --
password="$MYSQL_PASSWORD" "$MYSQL_DB" | awk -vFS="\t" 'NR > 1
{printf("%s ", $1);}'`

# Nacteni vsech zarizeni, na kterych maji byt monitorovana prenesena
data
SEZNAM_ZARIZENI=`echo -n "select id from zarizeni where
monitoring_klientu = 'Y' and monitoring_klientu_interval = \"$1\";"
| mysql -B -h "$MYSQL_HOST" -u "$MYSQL_USER" --
password="$MYSQL_PASSWORD" "$MYSQL_DB" | awk -vFS="\t" 'NR > 1
{printf("%s ", $1);}'`

# Pokud je perioda mensi nez 1 minuta, provedeme 60/n volani v
minute
     if [ "$1" -lt 60 ]; then
     for i in `seq 0 $1 59`; do

# Projdu vsechna mereni a pro vsechna pustim monitorovaci skript
          for ID in $SEZNAM_MERENI; do
               echo "Spoustim snmp mereni $ID - $i"
               php monitoring_snmp.php "$ID" &
          done

# Projdu vsechna zarizeni a pro vsechna pustim monitorovaci skript
          for DEV in $SEZNAM_ZARIZENI; do
               echo "Spoustim monitoring klientu $DEV - $i"
               php monitoring_klientu.php "$DEV" &
          done

          sleep $1
     done
else

# Pro ostatni periody provedeme volani pouze jednou (o zbytek se
stara cron)
# Spusteni vsech snmp mereni
     for ID in $SEZNAM_MERENI; do
          echo "Spoustim mereni $ID - jednorazove"
          php monitoring_snmp.php "$ID" &
     done

# Spusteni vsech monitoringu klientu
     for DEV in $SEZNAM_ZARIZENI; do
          echo "Spoustim monitoring klientu $DEV - jednorazove"
          php monitoring_klientu.php "$DEV" &
     done
fi
```

Obrázek 8: úryvek zdrojového kódu souboru backand.sh

5.3.3.1 Monitorování stavových hodnot zařízení

Monitorování stavových hodnot zařízení je realizováno dotazy na konkrétní SNMP OID řetězec. Dotaz je určen operačnímu systému RouterOS, který je nahraný na zařízení Mikrotik RouterBoard. Konkrétní identifikátory OID, které lze zadat pro tento typ měření v administračním webovém rozhraní, lze snadno přidat do databáze do tabulky „snmp_typy_mereni“. Tak se dá jednoduše dle požadavků uživatele manipulovat se schopnostmi celé aplikace. V databázi budou uloženy následující typy měření stavových hodnot zařízení:

- zatížení procesoru
- využité místo paměťového prostoru (datové úložiště zařízení)
- „uptime“ (doba od posledního restartu zařízení)

Tato měření je nutno aplikaci definovat v administrátorském webovém rozhraní. K zadávacímu formuláři se uživatel dostane kliknutím na odkaz SNMP u konkrétního zařízení v přehledu. Administrátorská sekce po zadání nového měření sama restartuje backend aplikace a od té chvíle běží monitorovací činnost.

Ke zjištění konkrétního SNMP OID řetězce lze v PHP použít funkcí *snmpget*. Následující ukázka zdrojového kódu tuto funkci demonstruje. Kód pochází ze souboru *monitoring_snmp.php*.

```
/* Vlastni cteni snmp hodnot ze zarizeni */

$res = snmpget($mereniR["ip"], $NETMON["SNMP_COMM"],
$mereniR["snmp_oid"]);

     if (!$res) { /* Nastala chyba */
          $chybovyStav = "Y";

          $hodnota = "";
}
     else { /* Nenastala chyba */
          $chybovyStav = "N";

          $hodnota = $res;
}
```

Obrázek 9: zdrojový kód pro vyčtení hodnot a ověření chyby

5.3.3.2 Monitorování přenesených dat klientů

Ke zjišťování informací o přenesených datech klientů je využit mechanizmus „/queue simple“ z operačního systému RouterOS zařízení Mikrotik. Tento mechanizmus sám uchovává na daném zařízení informace o počtu přenesených bajtů a paketů. Každá fronta (*queue*) je v aplikaci identifikována jednoznačným názvem. V projektu jsou jména front konstruována složením prefixu (Netmon-) a unikátním identifikátorem klienta z databáze.

Prefix je použit kvůli odlišení front definovaných uživatelem zvlášť přímo na zařízení od těch, které si aplikace sama vytváří. Na tyto hodnoty je možno se dotázat pomocí identifikátoru OID SNMP podstromů požadovaných objektů. Tato OID jsou v aplikaci nastavena v souboru *config.php*. V následující ukázce zdrojového kódu jsou vidět dotazy, které jsou zasílány zařízení pro měření přenesených dat klientů.

```
/* SNMP koren pro jmena queue simple */
$NETMON["SNMP_QUEUE_ROOT"]=".1.3.6.1.4.1.14988.1.1.2.1.1.2";


/* SNMP koren pro pocet uploadovanych bajtu */
$NETMON["SNMP_UPLOAD_ROOT"]=".1.3.6.1.4.1.14988.1.1.2.1.1.9";

/* SNMP koren pro pocet downloadovanych bajtu */
$NETMON["SNMP_DOWNLOAD_ROOT"]=".1.3.6.1.4.1.14988.1.1.2.1.1.8"
;

/* SNMP koren pro pocet uploadovanych paketu */
$NETMON["SNMP_UPLOAD_PACKETS_ROOT"]=".1.3.6.1.4.1.14988.1.1.2.
1.1.11";

/* SNMP koren pro pocet downloadovanych paketu */
$NETMON["SNMP_DOWNLOAD_PACKETS_ROOT"]=".1.3.6.1.4.1.14988.1.1.
2.1.1.10";
```

Obrázek 10: zdrojový kód pro dotazy na daná OID pro získání hodnot přenesených dat

Ke zjištění všech hodnot podstromu ze SNMP slouží v programovacím jazyce PHP funkce *snmpwalk*, jejíž návratovou hodnotou je pole SNMP objektů načtených ze zadaného podstromu. Její použití lze vidět na následujícím úryvku zdrojového kódu, pocházejícího ze souboru *monitoring_klientu.php*.

```
/* Funkce obalujici snmpwalk na zarizeni - kvuli debugovani */
function NETMON_snmpwalk($ip, $community, $oid, $timeout)
     {
          echo "$ip - $community - $oid - ";
          $res = snmpwalk($ip, $community, $oid, $timeout);

          echo "\n";
          return $res;
     }
```

Obrázek 11: zdrojový kód pro použití funkce snmpwalk

Použití výše uvedené funkce je demonstrováno na následujícím zdrojovém kódu, který rovněž pochází ze souboru *monitoring_klientu.php*. Výsledkem provedení následujícího příkladu, je vygenerování pěti polí s obsahem daných podstromů OID která jsou nastavena v souboru *config.php*. Tato pole jsou v souboru dále zpracovávána a ukládána do databáze.

```
/* Jmena queue simple */
$queueSimpleNames = NETMON_snmpwalk($zarizeniR["ip"],
$NETMON["SNMP_COMM"], $NETMON["SNMP_QUEUE_ROOT"],
$NETMON["SNMP_TIMEOUT"]);

/* Bajty upload */
$uploadBytes = NETMON_snmpwalk($zarizeniR["ip"],
$NETMON["SNMP_COMM"], $NETMON["SNMP_UPLOAD_ROOT"],
$NETMON["SNMP_TIMEOUT"]);

/* Bajty download */
$downloadBytes = NETMON_snmpwalk($zarizeniR["ip"],
$NETMON["SNMP_COMM"], $NETMON["SNMP_DOWNLOAD_ROOT"],
$NETMON["SNMP_TIMEOUT"]);

/* Pakety upload */
$uploadPackets = NETMON_snmpwalk($zarizeniR["ip"],
$NETMON["SNMP_COMM"], $NETMON["SNMP_UPLOAD_PACKETS_ROOT"],
$NETMON["SNMP_TIMEOUT"]);

/* Pakety download */
$downloadPackets = NETMON_snmpwalk($zarizeniR["ip"],
$NETMON["SNMP_COMM"], $NETMON["SNMP_DOWNLOAD_PACKETS_ROOT"],
$NETMON["SNMP_TIMEOUT"]);
```

Obrázek 12: zdrojový kód pro kód pro vygenerování pěti polí s obsahem podstromů OID

Po načtení všech informací z SNMP nutných pro monitorování klientů je nutno provést vyresetování čítačů v *queue simple*. Tím máme zajištěno, že při dalším měření dostaneme hodnoty přenesených dat a paketů přesně za časový úsek, který je dán periodou měření. K vynulování čítačů bylo použito aplikační rozhraní Mikrotik API. Klientskou implementaci v jazyce PHP dává k dispozici přímo výrobce na svých internetových stránkách (*http://wiki.mikrotik.com/wiki/API_PHP_class*). Jedná se o třídu, která realizuje samotnou komunikaci se zařízením a vrací výsledky volání API funkcí. Tato třída je uložena v souboru *mikrotik_api.php*. Vlastní funkce realizující uvedení čítačů zpátky do nulových hodnot (reset) je uvedena v následující části zdrojového kódu. Nachází v souboru *monitoring_klientu.php*.

```
/* Provedeni resetu counteru na queue simple na zarizeni */
function NETMON_reset_counters($ip, $user, $password)
     {
          echo "$ip - $user - $password\n";
          $API = new routeros_api();
          if ($API->connect($ip, $user, $password)) {
               $API->write("/queue/simple/reset-counters-all");
               $res = $API->read();
               return true;
          }
          else {
               return false;
          }
```

Obrázek 13: zdrojový kód pro uvedení čítačů do nulových hodnot (reset)

Využití uvedené funkce je demonstrováno na dalším úryvku zdrojového kódu ze souboru *monitoring_klientu.php*. Zde je zajištěno ověření návratové hodnoty funkce *NETMON_reset_counters* k detekci chyby zařízení.

```
/* Provedeni resetu counteru na zarizeni */

$resetCounters = NETMON_reset_counters($zarizeniR["ip"],
$zarizeniR["uzivatel"], $zarizeniR["heslo"]);

     if (!$resetCounters) {
          echo "Doslo k chybe pri reset counter !\n";
     }
```

Obrázek 14: zdrojový kód - ověření návratové hodnoty resetu - detekce chyby

5.4 Úkony na zařízení pro správnou funkci monitorovací činnosti

Před jakoukoliv manipulací se zařízením a především operačním systémem RouterOS je zapotřebí provést několik úkonů, aby výsledná aplikace se zařízením a systémem RouterOS mohla komunikovat. Nejprve je třeba povolit a nastavit na zařízení využívání protokolu SNMP, který zde bude sloužit jako prostředník mezi vyvíjenou aplikací a zařízením. To provedeme zadáním příkazu „**/snmp set enabled=yes**“ do příkazové řádky zařízení Mikrotik. Dále je třeba přidat SNMP komunitu, kterou využívá backend pro monitorování. Vytvoření komunity zajišťuje příkaz „**/snmp comunity add name=NETMON adress=0.0.0.0/0 security=none read-access=yes**“. Jako poslední úkon který je třeba provést je povolení aplikačního rozhraní API. To nám bude sloužit k tomu, aby uživatel výsledné aplikace mohl přidávat klientské stanice, které budou určeny pro monitoring. Navíc povolení API bude vyžadovat situace, kdy bude třeba uvést čítače do nulových hodnot (*reset counters*). Rozhraní povolíme opět příkazem v příkazové řádce „**/ip service enable api**“.

5.5 Výstupy monitorování přenesených dat klientů

Výstupem celé aplikace jsou grafy, které uživateli umožňují objektivně posoudit vytíženost sítě a monitorovat stav zařízení na kterém je prováděno měření. Díky grafickému zobrazení stavových hodnot, jako je vytížení procesoru nebo využité místo v paměti, může administrátor posoudit zatížení zařízení a zvážit nahrazení přístupového bodu výkonnějším modelem. To zvyšuje efektivitu a zajišťuje optimalizaci sítě. U každého monitorovaného klientského zařízení se zobrazí celkem čtyři grafy. Jedná se o grafy přenesených dat a paketů ve směru vzestupném i sestupném a to z hlediska klientského zařízení.

Všechny grafy, jak stavových hodnot, tak přenesených dat, jsou generovány pomocí knihovny JPgraph, která je volně dostupná a umožňuje uživateli využít téměř sedmdesát typů grafů. Jedná se o výkonný nástroj psaný v jazyku PHP, určený především pro generování grafických znázornění. Pomocí konfiguračních souborů lze jednoduše nastavit parametry grafu, jako jsou například: typ, měřítka, velikost, popisy os nebo barevné schéma (křivka, pozadí, popisy) a mnoho jiných estetických či funkčních prvků. Skriptu, generujícímu graf je také nutno nadefinovat data, která jsou určena pro grafické vykreslení. V tomto případě se jedná o čas na ose X a přenesená data resp. pakety na ose Y. Tyto informace jsou ukládány do databáze MySQL.

```
/* Zakladni parametry grafu */
      $graph = new Graph(1200,400,"auto");
      $graph->SetScale("textlin");
      $lineplot=new LinePlot($ydata);
      $lineplot->SetColor("blue");
      $graph->Add($lineplot);

      /* Nastaveni popisek osy X v grafu */
      $graph->xaxis->SetTickLabels($datax);
      $graph->xaxis->title->Set("usek = {$_GET["perioda_sekundy"]} sekund");

      /* Nastaveni popisku osy Y */
      if ($_GET["typ_grafu"] == "klient") { /* Graf klienta */
            $ypopisek = "";
            switch($_GET["kategorie_grafu"]) {
                  case "upload": $ypopisek = "kilobyte"; break;
                  case "download": $ypopisek = "kilobyte"; break;
                  case "upload_pakety": $ypopisek = "packet"; break;
                  case "download_pakety": $ypopisek = "packet"; break;
            }
            $graph->yaxis->title->Set($ypopisek);
```

Obrázek 15: zdrojový kód - nastavení parametrů grafu

Předcházející úryvek kódu zajišťuje nastavení několika základních parametrů grafu. Jedná se rozměry, barvu křivky, typ grafu a také definice popisků obou os. Na ose X jsou uvedeny hodnoty period v sekundách. Pro osu Y jsou definovány dva popisky, jelikož pro každé klientské zařízení monitorujeme kromě přenesených kilobajtů také přenesené pakety.

Následující část skriptu slouží ke korektnímu načtení vstupních hodnot, ze kterých následně po aplikaci knihovny JPgraph bude zhotoven graf v podobě vygenerovaného souboru formátu PNG. Ten je pak jednoduše umístěn na požadované místo v uživatelském

rozhraní. Nejprve se dle zadaného parametru v uživatelském rozhraní připraví data pro graf. Pro osu X se jedná o časové periody a pro osu Y se hodnoty z databáze volí podle toho, jestli se jedná o měření přenosů dat klienta nebo měření stavových hodnot.

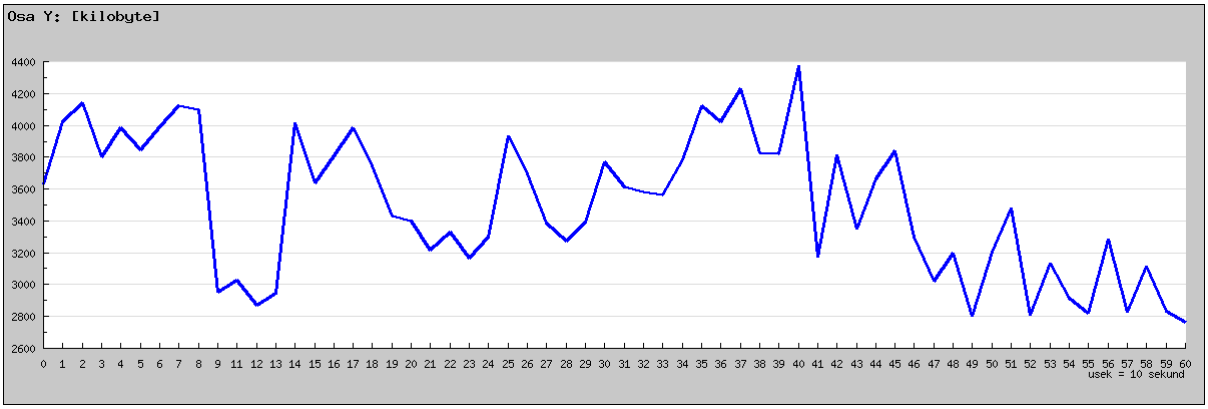
```
/* Podle parametru z url pripravim data pro graf - popisky na ose X */
/* Hodnoty na ose x jsou minuty podle periody v sekundach - 10s perioda = 10
minut se zobrazi na grafu 60s perioda = 60 minut se zobrazi na grafu */
      if (!isset($_GET["perioda_sekundy"])) {
            die ("Chyba");
      }

      $datax = array();
      for ($i = 0; $i < $_GET["perioda_sekundy"]; $i++) {
            $datax[$i] = $i;
      }

      /* Podle zadaneho klice a typu grafu nactu z databáze data pro osu Y */
      if (!NETMON_databaze()) {
            die ("Chyba databaze !");
      }

      if (!isset($_GET["typ_grafu"]) || !isset($_GET["kategorie_grafu"]) ||
      !isset($_GET["id"])) {
            die ("Chyba - nutno zadat vsechny parametry !");
      }

      if ($_GET["typ_grafu"] == "klient") { /* Zobrazeni grafu pro statistiku
      klienta */
      $ydata = NETMON_get_y_data($_GET["kategorie_grafu"], $_GET["id"],
      $_GET["perioda_sekundy"]);
      }

      elseif ($_GET["typ_grafu"] == "snmp") { /* Zobrazeni grafu mereni
      stavovych hodnot SNMP */

      $ydata = NETMON_get_y_snmp_data($_GET["id"], $_GET["perioda_sekundy"]);
```

Obrázek 16: zdrojový kód pro načtení hodnot pro generování grafu

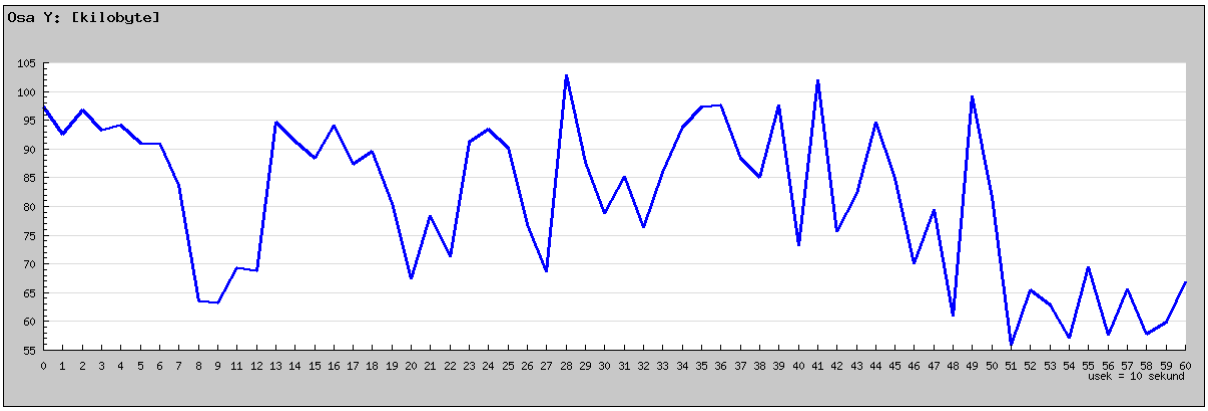
Na následujících obrázcích je vidět přehledný grafický výstup monitorovaných dat a paketů a stavové hodnoty vytížení procesoru. Časová osa grafů byla záměrně zvolena delší, aby uživatel mohl objektivně posoudit delší sledovaný časový úsek. Každý dílek časové osy X představuje jednu periodu měření. Celkem lze tedy sledovat šedesát period měření. Zvolíme-li periodu měření například třicet minut, můžeme posoudit obousměrný provoz za více jak 24 hodin. Je však nutno podotknout, že se zvětšující se periodou měření lze předpokládat zmenšující se přesnost měření. Pro naše účely je tedy vhodnější volit menší periody. Zvolíme-li periodu měření například 20 sekund, lze si lehce udělat obrázek o tom, kolik dat bylo během této periody přeneseno. U hodinové periody si jen těžko můžeme představit, kdy během této periody se přenášely data a kdy nikoliv.

Graf znázorňující vytížení procesoru přístupového bodu slouží správci sítě spíše pro orientační posouzení výkonnostních schopností zařízení. Kdyby se křivka pohybovala blízko hranici sto procent, a to konstantně, měl by se bezdrátový prvek nahradit výkonnějším modelem. Pro posouzení stability může uživatel využít monitorování a grafické znázornění tzv. *doby uptime*. Z grafu lze vidět, jak dlouho je zařízení zapnuto, a kdy došlo k výpadkům

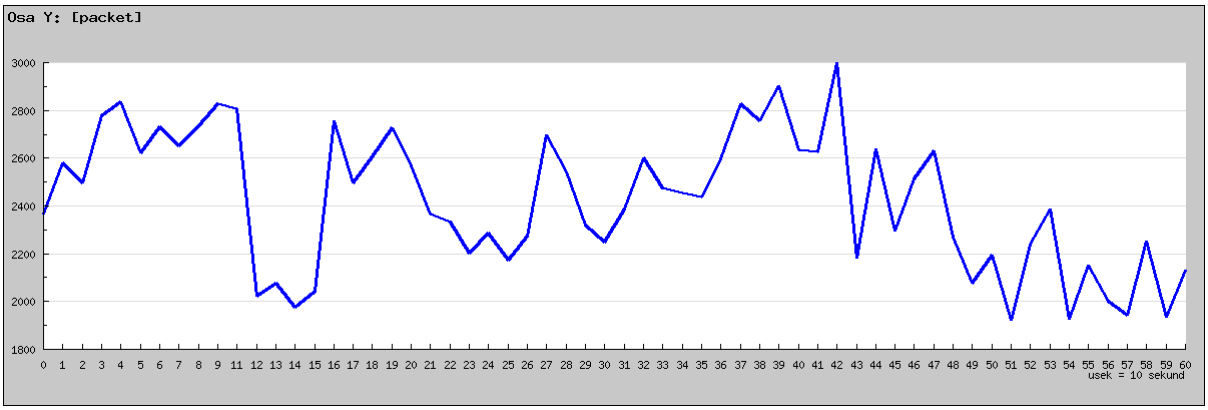
plánovaným či neplánovaným. Ze stavových hodnot lze dále sledovat obsazenost paměťového prostoru bezdrátového prvku. Toto měření rovněž slouží k optimalizaci sítě. Graf uživateli prezentuje, kolik paměti je ještě volné. Paměť se zaplňuje s přibývajícími záznamy ve směrovacích tabulkách, bráně firewall či s instalací nových aktualizací. Pokud by se křivka blížila hranici hodnoty velikosti paměti, bylo by nutné zvážit nahrazení bezdrátového prvku zařízením s větší pamětí.



Obrázek 17: monitorování přenesených bajtů - DOWNLOAD



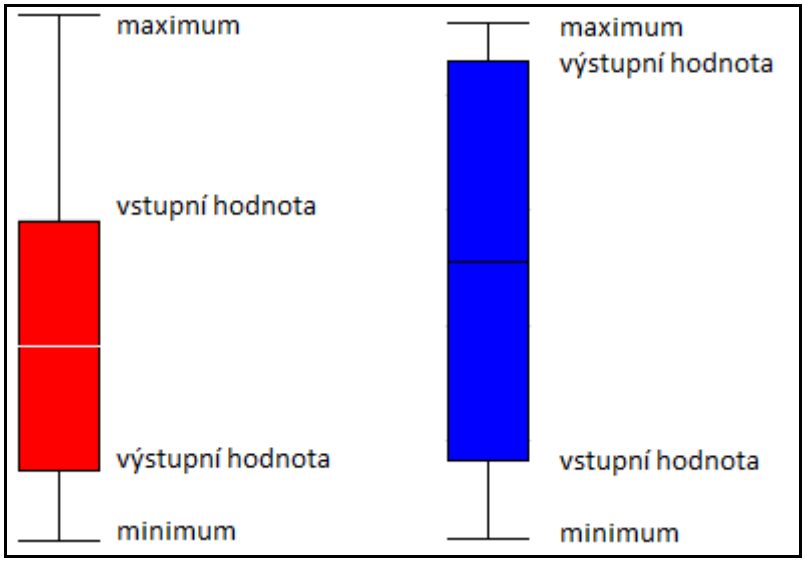
Obrázek 18: monitorování přenesených bajtů - UPLOAD



Obrázek 19: monitorování přenesených paketů - DOWNLOAD

5.6 Funkce pro pokročilé zpracování získaných dat I

K objektivnímu posouzení datových přenosů můžou posloužit i některé funkce pokročilého zpracování dat. V tomto případě se jedná o statistické vyhodnocení určitých, předem definovaných úseků dat, ze kterých bude funkce hledat maximum, minimum a zaznamenávat se budou také hodnoty vstupní a výstupní pro každý interval. Jako přehledný výstup byl zvolen krabicový graf, který nám umožňuje zahrnout všechny zmíněné parametry do přehledných obrazců, svící. Důležité je podotknout, že každá výstupní hodnota v grafickém znázornění je zároveň rovna vstupní hodnotě následujícího intervalu, následující krabici. Jednotlivé krabice tedy na sebe navazují. Na základě těchto parametrů může uživatel rychle posoudit, kde se přibližně hodnota měření pohybovala a jaké maximální a minimální hodnoty bylo v průběhu měření dosáhnuto. Zároveň je možné si představit průběh a chování přenosu. Výstupem tohoto statistického zpracování je tedy krabicový graf o třicetidvou krabicích, který je pro tyto účely určen. Na následujícím obrázku lze vidět popis jednotlivých pozic každé krabice.



Obrázek 20: vysvětlení parametrů svíce

V krabicovém grafu jsou barevně odlišeny situace, kdy vstupní hodnota je větší resp. nižší než vstupní. Každá krabice prezentuje určitou množinu hodnot z databáze, která je definovatelná a to 10, 20, 40, 60, 80, 100. Krabicový graf tak může popisovat až sto hodnot z databáze. Ve výsledku to znamená, že od chvíle zahájení měření bude každých až 100 změřených hodnot bráno jako celek, který poslouží pro vykreslení jedné svíce. Pro přesnější měření je vhodné volit kratší intervaly.

V některých případech grafického znázornění může docházet k situacím, které mohou vést k nepřehlednosti grafu v daném intervalu. Jednou z možných situací je, že jednotlivé parametry krabice mohou na výstupu splývat. V takovém případě je nutné si představit, jaké parametry se takto chovají a co nám tento fakt vypovídá o chování přenosu.

Celkem třicet krabic již může vypovídat o tom, jestli se jednalo o přenos spíše sporadický nebo se přenos choval konstantně, například přenos souboru o větší velikosti. Tato metoda statistického vyhodnocování se často používá ve všech možných oborech, zejména matematických a ekonomických (burza). Pro uživatele představuje tento způsob efektivní a rychlý nástroj pro zjištění potřebných informací o chování určitého jevu v minulosti a ke krátké předpovědi chování v budoucnosti.

```
/* Kontrola zadanych parametru - id a kategorie grafu musi byt nastaveny */

      if (!isset($_GET["id"]) || !isset($_GET["kategorie_grafu"]) ||
      !isset($_GET["perioda_sekundy"]) || !isset($_GET["pocet_hodnot"])) {
            die ("Chybne parametry!");
            }

/* Podle zadaneho klice a druhu grafu nactu z databaze data pro osu Y */
      if (!NETMON_databaze()) {
            die ("Chyba databaze !");
            }

/* Zakladni parametry grafu */
      $graph = new Graph(1200,400,"auto");
      $graph->SetScale("textlin");
      $kr = new BoxPlot(NETMON_get_boxplot_data($_GET["id"],
$_GET["kategorie_grafu"], $_GET["perioda_sekundy"],
$_GET["pocet_hodnot"]));
      $kr->SetWidth(40); /* Sirka krabice v pixelech */
      $kr->SetCenter(); /* Vycentrovat krabicove intervaly */
      $kr->SetColor("black", "blue", "black", "red");

/* Popis osy x */
      $velikost_useku = $_GET["perioda_sekundy"] * $_GET["pocet_hodnot"];
      $graph->xaxis->title->Set("usek = $velikost_useku sekund");

/* Nadpis grafu */
      $graph->title->Set("Osa Y: [kilobyte]");
      $graph->title->SetAlign("left");
```

Obrázek 21: zdrojový kód pro generování krabicového grafu

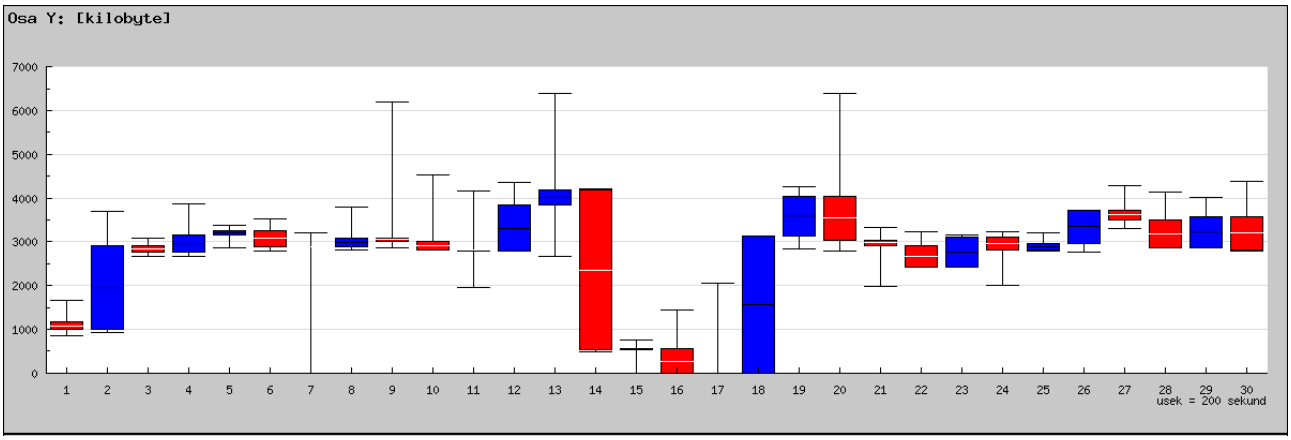
Uvedený úryvek kódu popisuje generování krabicového grafu. Jeho úkolem je generovat graf určitých rozměrů s danými popisy os a o definovaných barvách jednotlivých krabic. Uvedený kód široce souvisí se souborem *design.php* kde jsou uvedeny matematické a další funkce sloužící pro získání dat z databáze pro správné vykreslení požadovaných grafů. Níže je uvedena část kódu tohoto souboru, která se stará o potřebné výpočty hodnot které se mají vykreslit ve výsledném krabicovém grafu. Jedná se o zjištění minima, maxima, vstupní a výstupní hodnoty. Aby se graf vykreslil korektně, je nutné vložit i funkci pro zjištění průměru. V tomto případě průměru vstupní a výstupní hodnoty.

```
/* Inicializace hodnot pro vykresleni ve grafu */
            $kr_min = 0;
            $kr_max = 0;
            $kr_avg = 0;
            $kr_bottom = 0;
            $kr_top = 0;
```

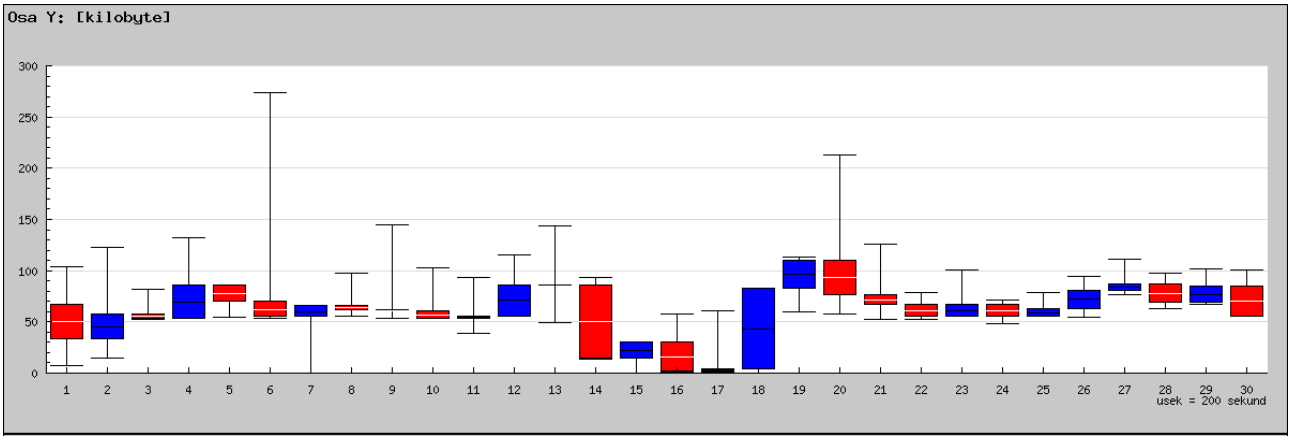
```
/* Pokud byla nactena nejaka data, provede se vypocet */
            if (count($hodnoty) != 0) {
/* Vypocet minima a maxima */
            $kr_min = min($hodnoty);
            $kr_max = max($hodnoty);

/* Dosazeni vstupni a vystupni hodnoty */
            $kr_top = $hodnoty[0];
            $kr_bottom = $hodnoty[count($hodnoty) - 1];

/* Prumer se vypocita ze vstupni a vystupni hodnoty */
            $kr_avg = (int)(($kr_top + $kr_bottom) / 2);
```

Obrázek 22: zdrojový kód pro výpočet hodnot pro krabicový graf

Výstupem první části pokročilého zpracování získaných dat jsou přehledné krabicové grafy, které uživateli usnadňují pochopení chování přenosů jednotlivých klientů v daných časových periodách. Je však důležité si uvědomit význam barev a velikost jednotlivých svící.



Obrázek 23: krabicový graf - DOWNLOAD



Obrázek 24: krabicový graf - UPLOAD

5.7 Funkce pro pokročilé zpracování získaných dat II

V další části pokročilého zpracování dat je pojednáno o statistickém zpracování dat pomocí prostého klouzavého a exponenciálního klouzavého průměru. Tyto dva parametry nám ulehčují pochopit chování přenosů dat v průběhu času měření.

Klouzavý průměr popisuje průměrnou hodnotu údajů v šíři daného okna. Například pěti hodnotový klouzavý průměr značí průměrnou hodnotu za posledních pět hodnot apod. Když propojíme hodnoty v grafu, získáme křivku klouzavého průměru. Velikost prostého klouzavého průměru závisí na dvou parametrech, průměrovaných hodnotách a šířce okna, v rámci kterého se bude průměr počítat.

Použití prostého klouzavého průměru je vhodné, jelikož jeho výpočet je jednoduchý. Na druhou stranu má však tento průměr jeden zásadní nedostatek. Jedná se o to, že se změnou hodnoty se změní dvakrát. Nejprve jakmile se v definovaném okně objeví nový údaj. To je vhodné, jelikož průměr nám bude popisovat vývoj hodnoty. Problém však je, že se prostý průměr změní ještě jednou a to v situaci, kdy se na konci časového okna objeví jiná hodnota, stará je z okna odstraněna.

Druhou možností sledování průběhu je exponenciální pohyblivý průměr. Je vhodnějším nástrojem pro sledování chování hodnot. Vkládá větší váhu aktuálním hodnotám a rychleji reaguje na závislosti na starých hodnotách. Vhodné je, že exponenciální klouzavý průměr neskáče nahoru a dolu tak jako prostý klouzavý průměr. U této verze průměru je krátké okno citlivější na změny hodnot. Často může měnit tvar a vytváří ho pilovitý. Exponenciální klouzavý průměr se v našem případě počítá podle následujícího vzorce.

$$\varnothing_{\exp} = S * K + P_{\varnothing \exp} * (1 - K)$$

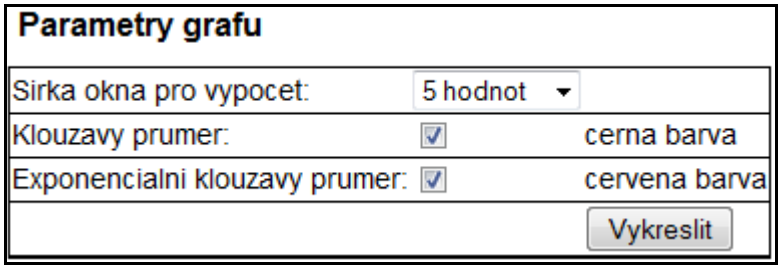
$$S = současná\ hodnota$$

$$P_{\varnothing \exp} = předposlední\ hodnota\ průměru$$

$$N = počet\ hodnot, za\ něž\ počítáme\ průměr$$

$$K = \frac{2}{N+1}$$

Do aplikace jsou průměry vloženy jako dodatečné funkce, které si uživatel může zapnout. Lze zvolit velikost okna 5-10 hodnot. Prostý průměr je označen černou a exponenciální průměr barvou červenou.



Obrázek 25: nastavení parametrů

```
/* Pokud je vyzadovan klouzavy prumer a je nastavena sirka okna, tak data
prozeneme funkci pro vygenerovani dat pro krivku klouzaveho prumeru a tuto
krivku rovnou pridame do grafu */
      if (isset($_GET["mavg"]) && $_GET["mavg"] == "on" &&
            isset($_GET["okno"]) && is_numeric($_GET["okno"])) {
            $mavg_data = NETMON_gen_moving_average($ydata, $_GET["okno"]);
            $mavg_lineplot = new LinePlot($mavg_data);
            $mavg_lineplot->SetColor("black");
            $graph->Add($mavg_lineplot);
      }
/* Pokud je vyzadovan exponencialni klouzavy prumer a je nastavena sirka
okna, tak data prozeneme funkci pro vygenerovani dat pro krivku
exponencialniho klouzaveho prumeru a tuto krivku rovnou pridame do grafu */
      if (isset($_GET["emavg"]) && $_GET["emavg"] == "on" &&
            isset($_GET["okno"]) && is_numeric($_GET["okno"])) {
            $emavg_data = NETMON_gen_exp_moving_average($ydata,
            $_GET["okno"]);s
            $emavg_lineplot = new LinePlot($emavg_data);
            $emavg_lineplot->SetColor("red");
            $graph->Add($emavg_lineplot);
```

Obrázek 26: zdrojový kód pro vkládání průměrů do grafu s průběhy

Výše uvedená ukázka zdrojového kódu skriptu *graf.php* prezentuje funkci pro vkládání prostého a exponenciálního klouzavého průměru do grafů s naměřenými průběhy.

```
/* Vygeneruje z pole data pro krivku klouzaveho prumeru */
      function NETMON_gen_moving_average($data, $okno)
      {
            $res = array();
            $sum = 0;

/* Projeti vsech prvku pole */
      foreach ($data as $key => $value) {

/* Pokud je index v poli mensi nez sirka okna - 1, tak pouze pricitame do
sumy a do pole dame nulu */
      if ($key < ($okno - 1)) {
            $sum += $value;
            $res[] = "x"; /* x je specialni znak pro jpgraph, ktery mu
rekne, aby danou hodnotu nevykresloval */
            }
      else {

/* Odecteme od sumy hodnotu, ktere je mimo okno a nepocita se s ni a
pricteme normalni hodnotu v dane iteraci a vypocteme z ni prumer a pridame
do pole, ktere se bude vracet */
            $sum -= $data[$key - $okno];
            $sum += $value;
            $res[] = (int)($sum / $okno);
```

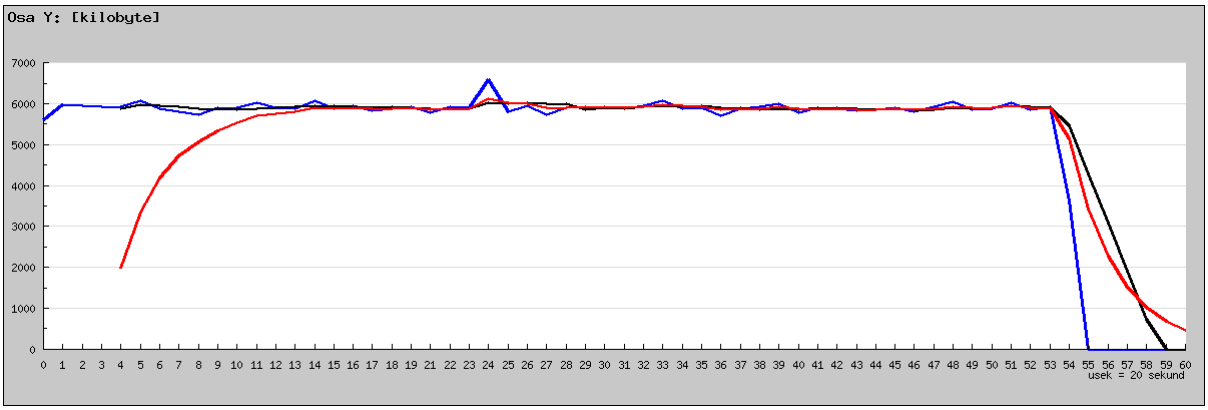
Obrázek 27: zdrojový kód pro získání dat pro klouzavý průměr

Druhá a třetí ukázka prezentují funkce, které dle zadaného okna pro výpočet klouzavých průměrů z databáze získají potřebné data. Ty funkce v první uvedené ukázce pomocí knihovny *jpgraph* vykreslí. Tato funkce je volána skriptem *graf.php*.

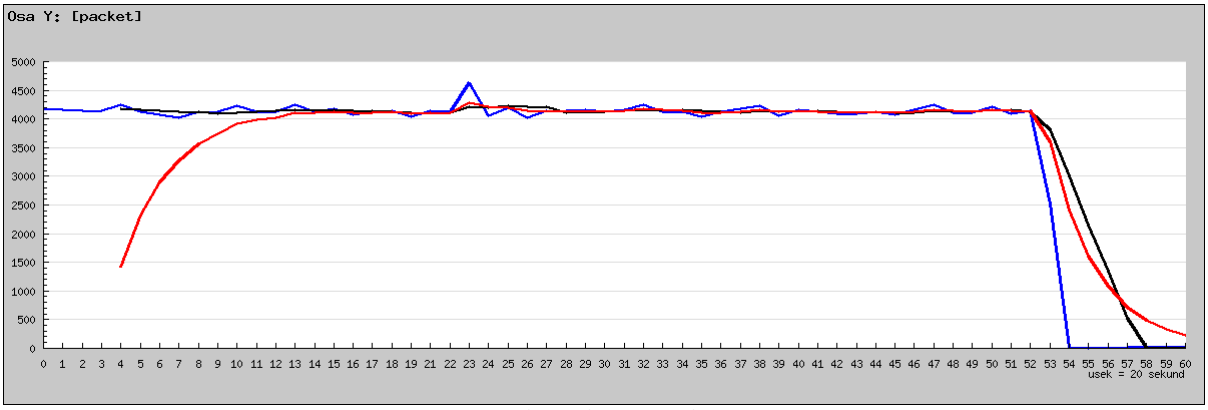
```
/* Vygeneruje ze zadaneho pole data pro krivku exponencialniho klouzaveho
prumeru */
            function NETMON_gen_exp_moving_average($data, $okno)
            $res = array();
            $sum = 0;
/* Konstanta pro vzorec vypoctu exponencialniho klouzaveho prumeru */
            $k = 2 / ($okno + 1);
/* Projeti vsech prvku pole */
            $lastVal = 0;
            foreach ($data as $key => $value) {
      $exp_avg = (int)($value * $k + $lastVal * (1 - $k)); /* Vypocet exp.
klouzaveho prumeru */
      $res[] = $exp_avg; /* Pridani do pole, ktere bude funkce vracet */
      $lastVal = $exp_avg; /* Hodnota uchovana pro dalsi iteraci */
```

Obrázek 28: zdrojový kód pro získání dat pro exponenciální klouzavý průměr

Výstupem první části statistického zpracování získaných dat je následující graf prezentující monitoring přenesených dat při periodě 20 sekund. V grafu průběhů jsou zakresleny oba typy zmíněných klouzavých průměrů.



Obrázek 29: vykreslení průměrů do grafů – bajty – DOWNLOAD



Obrázek 30: vykreslení průměrů do grafů – pakety – DOWNLOAD

5.8 Funkce pro pokročilé zpracování získaných dat III

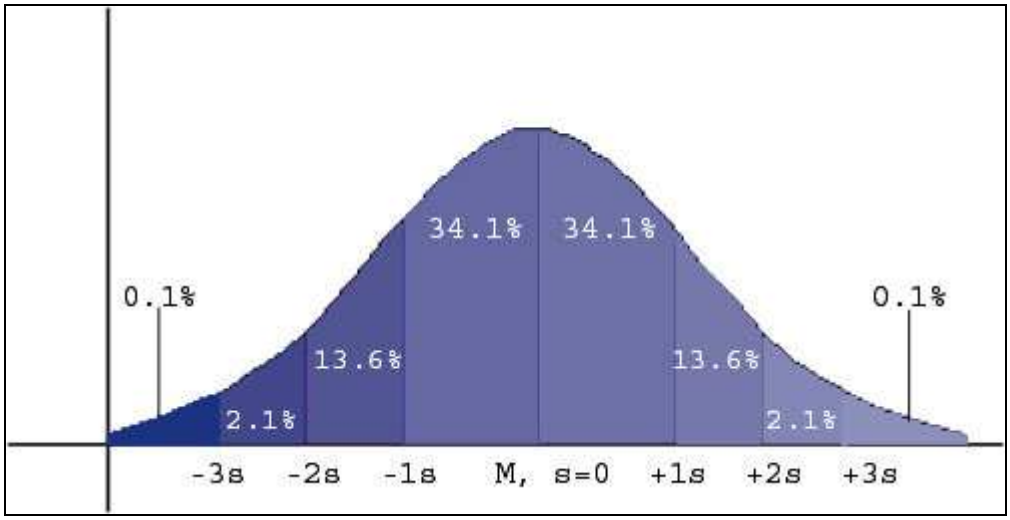
V dalším bodě statistického zpracování se diplomová práce zabývá implementací burzovních nástrojů oscilátoru Stochastic a indikátoru volatility Bollinger Bands.

O oscilátoru Stochastic můžeme říci, že se jedná o indikátor hybnosti. V oblasti burzy a obchodování s cennými papíry ukazuje umístění aktuální zavírací ceny vztažené k rozpětí nejvyšší a nejnižší ceny v průběhu několika období. Zavírací úrovně situovány blízko vrcholu rozpětí indikují nákupní tlak a naopak ty, které se nachází blízko dna rozpětí, indikují prodejní tlak. Pro naše účely, tedy účely statistického zpracování přenosů, je však nutno podotknout několik faktů. Základem zhodnocení pomocí stochastického oscilátoru je prvotní, základní interval, jelikož ve vzorci počítáme i s poslední hodnotou předcházejícího intervalu, tvz. uzavírací cenou v oblasti burzovní. Krom toho je třeba ještě si ujasnit, jak velké bude okno hodnot, se kterými budeme pro daný interval pracovat a okno intervalů, ze kterých budeme hodnoty pro výpočet %K získávat. Do vzorce tedy dosazujeme nejvyšší a nejnižší hodnotu z *n* intervalů a nakonec celý výraz vynásobíme stem. Zjednodušeně řečeno dosazujeme nejvyšší z nejvyšších hodnot a nejnižší z nejnižších hodnot. Výsledek je procentuální hodnota. [7]

$$\%K = \left(\frac{S_{end} - S_{\min}}{S_{\max} - S_{\min}} \right) \cdot 100$$

Ve výsledku křivka %K může mít v některých případech nežádoucí odchylky. Většinou je tedy doprovázena další křivkou, kterou označujeme %D. Jedná se o klouzavý průměr vypočtených hodnot K. Standarně se průměr počítá z posledních třech hodnot K.

Dalším pozoruhodným nástrojem pro statistické vyhodnocování jsou Bollingerovy čáry, rovněž často využívané v oblasti burzy. Využití je velmi prospěšné a pestré, avšak je nutné chápat základní podstatu. Tento indikátor byl vynalezen v roce 1980 obchodníkem Johnem Bollingerem. Vzniku předcházely hluboké znalosti statistických teorií a metod. Nejzákladnější a nejdůležitější myšlenkou je teorie statistické distribuce. Ta říká, že 95% dat v definovaném intervalu jsou v rozpětí dvou standardních odchylek. Již v minulosti bylo dokázáno, že pokud máme náhodná data, pak je možné tato data poskládat do grafické podoby zvonu, nebo také zvané „bell curve“. Zjednodušeně řečeno, kdybychom data interpretovali do grafické podoby a zaznamenávali na kostičkovaný papír, distribuce dat by se dala jednou čárou popsat jako „bell curve“, nebo-li podoba "zvonu". Tato křivka se dále dělí na několik částí: střed, nebo-li průměr z dat v daném časovém intervalu a dále na části zvané standardní odchylky od průměru. Krom toho platí několik pravidel. 68% všech dat v daném časovém intervalu se nachází v rozmezí -1s až +1s. Tomuto intervalu se říká první standardní odchylka. 95% dat se nachází v rozmezí -2s až +2s což je druhá standardní odchylka. Posledním pravidlem je, že 99,7% dat v daném časovém intervalu je v rozmezí -3s až +3s, neboli v intervalu třetích standardních odchylek. Lépe je vše zřejmé na obrázku. [7]



Obrázek 31: distribuce náhodných dat

Máme-li tedy takovouto distribuci náhodných dat a víme-li, že 95% dat se pohybuje mezi dvěmi standardními odchylkami, můžeme říci, co se stane v případě, že přijde hodnota, která se bude nalézat mimo tyto dvě standardní odchylky. Dojde k takzvanému návratu k průměru, trh se otočí a vrátí zpět k průměrné ceně. Toto je celá podstata indikátoru Bollinger Bands. Zjednodušeně řečeno, v případě že se hodnota dostane nad nebo pod křivku, následně se hodnota zase vrátí k průměru. Bollingerův pás je tedy výborný signalizátor, který nás upozorňuje na změnu, a navíc lze krátkodobě předpovědět, jak se bude graf a tedy i hodnoty dále pohybovat.

Graf Bollingerova pásu se většinou slučuje s grafem svícovým (krabicovým), aby bylo jasně vidět, kdy se hodnota dostane nad a pod křivku. Pás se skládá celkem z tří křivek. Prostřední křivkou je klouzavý průměr a další dvě křivky dostaneme, když k průměru přičteme, resp. odečteme vypočtenou hodnotu. Výpočet hodnot se zakládá na následujících vzorcích. [7]

Výpočet klouzavého průměru (moving average): $$MA = \frac{\sum_{i=1}^{n} y_i}{n}$$

Výpočet horní, resp. dolní hranice pásu: $$BB_{h/d} = MA \pm D\sqrt{\frac{\sum_{i=1}^{n}(y_j - MA)^2}{n}}$$

D….počet standardních odchylek
n…..počet hodnot, y_j…..hodnoty

Nástroj Bollinger Bands nelze považovat za stoprocentní. Lze ho však použít pro krátkodobou predikci chování sledované funkce. V oblasti trhu je to velmi vážená pomůcka pro obchodování. Ze všeho nejdůležitější je však před použitím pochopit podstatu indikátoru volatility.

Následující ukázka je úryvek zdrojového kódu, který prezentuje část funkce netmon_gen_boll_data, která slouží pro vygenerování polí hodnot křivek indikátoru Bollinger Bands podle vzorečků uvedených výše. Funkce zjistí data, která jsou určena pro vykreslení křivky klouzavého průměru Bollingerova pásu a to i mimo jiné na základě uživatelsky nastaveného parametru okna. Dále je podstatné vypočítat směrodatnou odchylku, která hraje důležitou roli při získávání křivek tvořících hranice pásu Bollinger Bands. Vypočítá se tak, že se v každém intervalu náležícímu do definovaného okna odečte hodnota průměru intervalu a klouzavého průměru, následně se výsledky umocní na druhou, sečtou a nakonec podělí počtem hodnot a výsledek se odmocní. Nejdůležitější křivkou je klouzavý průměr, který v tomto případě vypočítáváme z *y* posledních výstupních hodnot jednotlivých intervalů, které představují svíce v krabicovém grafu. Toto okno je uživatelsky nastavitelné ve webovém prostředí aplikace. Dále funkce na základě vypočítaného klouzavého průměru a směrodatné odchylky již může vypočítat horní a dolní hranice Bollingerova pásu. K hodnotě průměru přičte, respektive odečte vypočítané odchylky na základě vzorce.

```
$res[0] = NETMON_gen_moving_average_boll($mavgData, $okno); /* Pridani
klouzaveho prumeru do vysledneho pole */

/* Vypocitani dat pro horni a dolni krivku */
            for ($i = 0; $i < count($res[0]); $i++) {

/* Pokud je generovana pozice mensi nez velikost okna, nezobrazi se nic */
            if ($i < $okno - 1) {

/* x - specialni znak pro jpgraph pro nevykreslovani krivky */
                        $res[1][$i] = "x";
                        $res[2][$i] = "x";
            }
            else {

/* Vypocitani smerodatne odchylky pro $i krok a pro nalezity pocet kroku do
historie a podle ni se ulozi hodnota pro horni a dolni krivku */
                        $num = 0;
                        for ($j = 0; $j < $okno; $j++)
                        {
/* Hodnota v danem bode minus hodnota MA v danem bode */
                        $tmpPart = $mavgData[$i - $j] - $res[0][$i - $j];
                        $tmpPart = $tmpPart * $tmpPart;
                        $num += $tmpPart;
                        }
                        $num = (int)($num / $okno);
                        $num = (int)(sqrt($num));
                        $num = $num * 1;

/* Ulozeni horni a spodni hodnoty pro krivky */
                        $res[1][$i] = $res[0][$i] + $num;
                        $res[2][$i] = $res[0][$i] - $num;
```

Obrázek 32: zdrojový kód pro výpočet hodnot pro indikátor volatility

Další úryvek kódu pochází ze souboru graf2.php a ukazuje způsob zobrazení křivek indikátoru Bollinger Bands. Jeho úkolem je vygenerování pole s hodnotami pro křivky indikátoru funkcí netmon_gen_boll_data, která již byla uvedena. Dále se stará o postupné zobrazení tří křivek indikátoru Bollinger knihovnou jpgraph. Je zde nastaveno jakou barvou se mají jednotlivé křivky vykreslovat a jakou mají mít šířku.

```
/* Je-li vyzadovan indikator Bollinger Bands, tak overime parametry a
zobrazime ho */
      if ($_GET["boll"] == "on" && is_numeric($_GET["okno"])) {

      /* Vygenerovani ciselnych dat pro krivky */
            $bollData = NETMON_gen_boll_data($krData, $_GET["okno"]);

      /* Krivka klouzaveho prumeru s nastavenou sirkou okna */
            $mavgPlot = new LinePlot($bollData[0]);
            $mavgPlot->SetColor("black");
            $mavgPlot->SetWeight(2);
            $graph->Add($mavgPlot);

      /* Krivka horni casti indikatoru Bollinger Bands */
            $highPlot = new LinePlot($bollData[1]);
            $highPlot->SetColor("blue");
            $highPlot->SetWeight(2);
            $graph->Add($highPlot);

      /* Krivka dolni casti indikatoru Bollinger Bands */
            $lowPlot = new LinePlot($bollData[2]);
            $lowPlot->SetColor("red");
            $lowPlot->SetWeight(2);
            $graph->Add($lowPlot);
      }
```

Obrázek 33: zdrojový kód pro zobrazení pásu Bollinger Bands

```
/* Vygenerujeme data pro oscilator stochastic */
      function NETMON_gen_stochastic($data, $okno)
      {
            $res = array();
/* Transformace pole pro pouziti ve vypoctech */
            $tmpArrayMin = array();
            $tmpArrayMax = array();
            $tmpArrayClose = array();
            for ($i = 0; $i < count($data); $i++) {
                  $openVal = $data[$i]; $i++;
                  $closeVal = $data[$i]; $i++;
                  $minVal = $data[$i]; $i++;
                  $maxVal = $data[$i]; $i++;
                  $avgVal = $data[$i];
                  $tmpArrayMin[] = $minVal;
                  $tmpArrayMax[] = $maxVal;
                  $tmpArrayClose[] = $closeVal;
```

```
/* Pro kazdou polozku v poli se vypocita hodnota stochastic */
            $absMin = 0; /* Absolutni minimum na okne n intervalu */
            $absMax = 0; /* Absolutni maximum na okne n intervalu */
            foreach ($tmpArrayClose as $key => $val) {

/* Pokud neni jeste dostatek hodnot, tak je hodnota 0 */
                  if ($key < $okno) {
                        $res[] = 0;
                        continue;
                  }
/* Vyberu z poslednich n hodnot maximum a minimum */
                  $absMin = $tmpArrayMin[$key];
                  for ($i = 0; $i < $okno; $i++) { /* Vybrani minima */
                        if ($tmpArrayMin[$key - $i] < $absMin) {
                              $absMin = $tmpArrayMin[$key - $i];
                        }
                  }
                  $absMax = $tmpArrayMax[$key];
                  for ($i = 0; $i < $okno; $i++) { /* Vybrani maxima */
                        if ($tmpArrayMax[$key - $i] > $absMax) {
                              $absMax = $tmpArrayMax[$key - $i];
                        }
                  }
/* Pokud je rozdil maxima a minima 0, tak je stochastic taky roven 0 -
osetreni kvuli deleni 0 */
                  if (($absMax - $absMin) == 0) {
                        $res[] = 0;
                        continue;
                  }
/* Vypocitani hodnoty stochasticu a jeho ulozeni do pole */
            $res[] = (int)((($val - $absMin) / ($absMax - $absMin)) * 100);
            }
            return $res;
      }
```

Obrázek 34: zdrojový kód pro generování oscilátoru Stochastic a provedení výpočtů

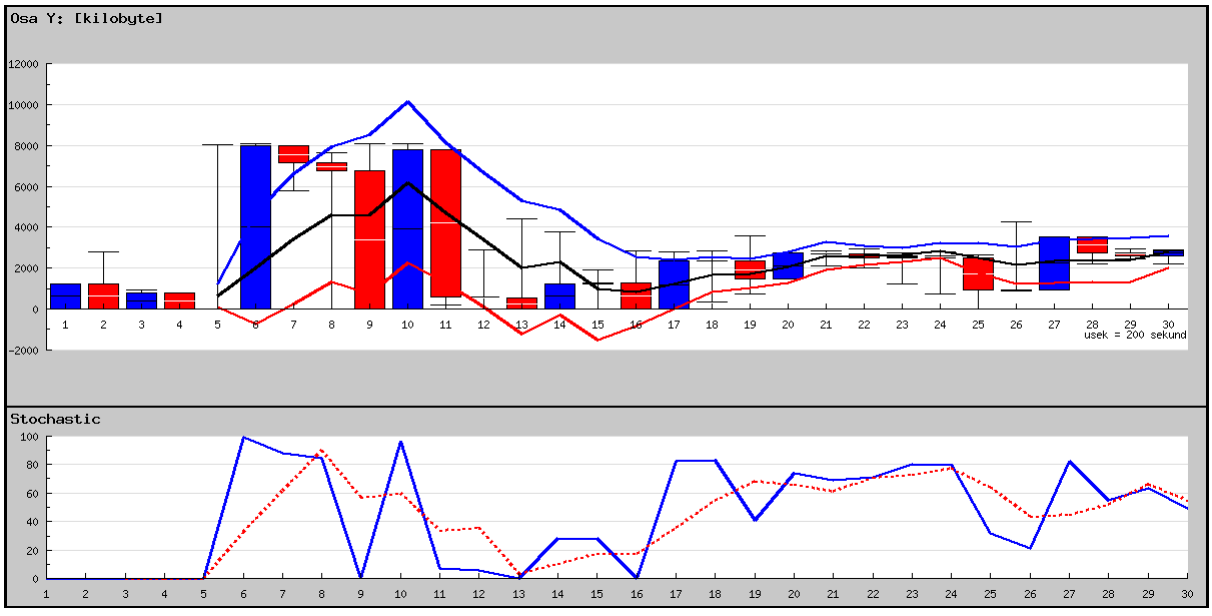
Předcházející dva úryvky kódu slouží pro vygenerování grafu oscilátoru Stochastic a pocházejí ze stejného souboru. Tato rozsáhlá funkce je volána ze souboru graf2.php. Funkce se nejprve postará o určení dat, které jsou pro graf podstatné a následně transformuje pole databáze tak, aby bylo možné provést potřebné výpočty. Z nastavených *n* intervalu funkce vybere absolutní minimum, tedy nejmenší hodnotu z nejmenších a stejně tak i hodnotu nejvyšší, tedy maximum. Důležité je i zmínit fakt, že v případě, že funkce nemá k dispozici dostatek hodnot, určí jako výchozí hodnotu nulu. Další ošetření aplikace spočívá v tom, že v případě že rozdíl získaných hodnot maximálního maxima a minimálního minima je roven nule, dosadí funkce rovnou za vypočítávanou hodnotu nulu. Mohlo by totiž dojít k situaci dělení nulou a tedy i výraznému zkreslení grafu nebo nefunkčnosti vykreslování. Posledním úkolem uvedené funkce je vypočítání hodnoty oscilátoru Stochastic dle uvedeného vzorce a uložení do pole hodnot v databázi.

Poslední ukázka týkající se čtvrtého bodu praktické části diplomové práce popisuje způsob, jakým se zobrazují křivky oscilátoru Stochastic. V tomto případě se totiž nevykresluje samostatný graf, ale graf indikátoru volatility se přidružuje ke grafu krabicovému. Je zde definováno jakou barvou a jakým typem křivky se bude křivka Stochastic a její klouzavý průměr zobrazovat. Pro oscilátor byla zvolena plná silnější čára a pro klouzavý průměr čára přerušovaná a slabší.

```
/* Vygenerovani grafu oscilatoru stochastic */
      $stoch1 = new Graph($graph_width,(int)($graph_height / 2),"auto");
      $stoch1->SetScale("textlin");
      $stochData = NETMON_gen_stochastic($krData, $_GET["stoch"]);
      $stoch1_kr1 = new LinePlot($stochData);
      $stoch1_kr1->SetColor("blue");
      $stoch1_kr1->SetWeight(2);
      $stoch1->title->Set("Stochastic");
      $stoch1->title->SetAlign("left");
      $stoch1->Add($stoch1_kr1);
/* Pridani do grafu stochastic klouzavy prumer */
      $stoch1_kr2 = new LinePlot(NETMON_gen_moving_average($stochData, 3));
      $stoch1_kr2->SetColor("red");
      $stoch1_kr2->SetWeight(2);
      $stoch1_kr2->SetStyle("dashed");
      $stoch1->Add($stoch1_kr2);
```

Obrázek 35: zdrojový kód pro zobrazení křivek oscilátoru Stochastic

Na následujícím obrázku je vidět grafický výstup pokročilého zpracování získaných dat pomocí oscilátoru Stochastic a indikátoru volatility Bollinger Bands ve směru příchozím. Jsou patrné změny chování Bollingerova pásu v průběhu měření v závislosti na množství přenášených dat.



Obrázek 36: grafický výstup Bollinger Bands a Stochastic

5.9 Krátkodobá predikce intenzity síťového provozu

Poslední kapitola praktické části se zaměřuje na využitelnost implementovaných funkcí pokročilého zpracování dat pro krátkodobou predikci intenzity síťového provozu. Predikce se v tomto případě zakládá na indikátoru volatility Bollinger Bands a oscilátoru Stochastic. Přesněji řečeno predikovaná maximální a minimální hodnota se zakládá na odchylce vypočítávané dle vzorce u indikátoru volatility a průměru předcházejících *n* intervalů. Predikovaná průměrná hodnota se odvíjí od průměru všech hodnot posledních *n* intervalů. Důležité je zmínit, že funkce predikuje jednu svíci v novém krabicovém grafu. Graf je situován těsně pod grafy krabicový a Stochastic viz. Obr.. Uživatel tak může lehce subjektivně posoudit věrohodnost predikce.

Zmíněnou funkci lze vidět v následujícím úryvku kódu. Jedná se o část funkce *netmon_gen_predict_data* ze souboru *design.php*. Funkce provádí nejprve výpočet velikosti pohybu hodnoty `$delta`, která je použita v případě signálu oscilátoru Stochastic pro růst nebo pokles. Vypočítá se průměrná hodnota přenosu pro každý z *n* intervalů, kde *n* představuje šířku okna u indikátoru volatility. Dále se určí minimální a maximální průměr ze zjištěných průměrů, provede se jejich rozdíl a následně se vydělí šířkou okna. Výsledkem této operace je zjištěná velikost pohybu hodnoty. Matematicky lze tuto operaci zapsat takto:

$$p_i = \frac{\sum_{j=1}^{r} y_j}{r}$$

$$d = \frac{p_{\max} - p_{\min}}{n} = \texttt{\$delta}$$

p_i …….průmer daného intervalu
r……počet hodnot v intervalu
y_j……..jednotlivé hodnoty v intervalu
p_{max} …….maximální průměr
P_{min} …….minimální průmer
n……..velikost okna

Funkce pokračuje výpočtem parametru `$absAvg`, což je hodnota klasického průměru z *n* posledních intervalů. Z této hodnoty se odvíjí předpověď průměrné hodnoty krabice. Důležitým momentem celé predikce je hodnota oscilátoru Stochastic, která zde hraje klíčovou roli. Je-li hodnota Stochastic menší než 25%, bude predikovaná průměrná hodnota rovna součtu parametrů `$absAvg + $delta`. Jinými slovy v tomto případě se bude očekávat nárůst. V případě že se hodnota Stochastic pohybuje v intervalu 25%-75% bude se předpokládat stagnace průměru, a tedy že bude roven parametru `$absAvg`. Poslední situací je hodnota Stochasticu větší než 75%. V takovém případě se bude očekávat pokles a predikovaná hodnota průměru tedy bude `$absAvg - $delta`. Na závěr funkce vypočte maximum a minimum predikované svíce přičtením resp. odečtením odchylky Bollingerova pásu od získaného průměru.

```
/* Projedu intervaly a pro kazdy vygeneruji predikci */
           foreach ($intervaly as $key => $interval) {
/* Pokud je index intervalu mensi nez hranice generovani, tak se negeneruje
nic, protoze nemam potrebna data z indikatoru */
                  if ($key < $minIndex) {
                        continue;
                  }
                  /* Stochastic hodnota */
                  $stochVal = $stochData[$key];

                  /* Odchylka z Bollingera */
                  $bollDev = $bollData[1][$key] - $bollData[0][$key];
/* Zjisteni intervalu, ktere se pouziji pro predikci prumery - jedna se o
aktualni interval ($key) a n intervalu dozadu podle sirky okna pouzite pro
vypocet Bollinger indikatoru */
                  $prumery = array();
                  for ($j = 0; $j < $okno; $j++) {
```

Obrázek 37: zdrojový kód pro generování predikce dle uvedených funkcí

```
/* Zjisteni prumerne hodnoty mereni */
                        $tmpInterval = $intervaly[$key - $j];
                        $sql = "select avg($kategorie) as hodnota from
monitoring_klientu_hodnoty where (cas >= '{$tmpInterval[0]}') and (cas <
'{$tmpInterval[1]}') and id_klienta = '$id_klienta'";
                        $avgQ = mysql_query($sql);
                        $avgR = mysql_fetch_array($avgQ);
                        $prumery[] = (int)($avgR["hodnota"]);
                  }
/* Vybrani minimalniho a maximalniho prumeru */
                  $minAvg = min($prumery);
                  $maxAvg = max($prumery);
                  $delta = (int)(($maxAvg - $minAvg) / $okno);
                  $absAvg = (int)(array_sum($prumery) / count($prumery));

                  $resValAvg = 0;

/* Signal k narustu dle stochasticu- novy prumer +- odchylka z Bollinger */
                  if ($stochVal <= 25) {
                        $resValAvg = $absAvg + $delta;
                  }
/* Signal stagnace dle stochasticu - +- odchylka z Bollinger  */
                  if ($stochVal > 25 && $stochVal < 75) {
                        $resValAvg = $absAvg;
                  }
/* Signal k poklesu dle stochasticu- novy prumer +- odchylka z Bollinger */
                  if ($stochVal >= 75) {
                        $resValAvg = $absAvg - $delta;
                  }
/* Vypocet minima a maxima podle prumeru a odchylky z Bollinger Bands */
                  $resValAvg = (int)($resValAvg / 1024);
                  $resValMin = $resValAvg - $bollDev;
                  $resValMax = $resValAvg + $bollDev;

/* Vygenerovani predikovane krabice */
                  $result[] = $resValAvg;
                  $result[] = $resValAvg;
                  $result[] = $resValMin;
                  $result[] = $resValMax;
                  $result[] = $resValAvg;
```

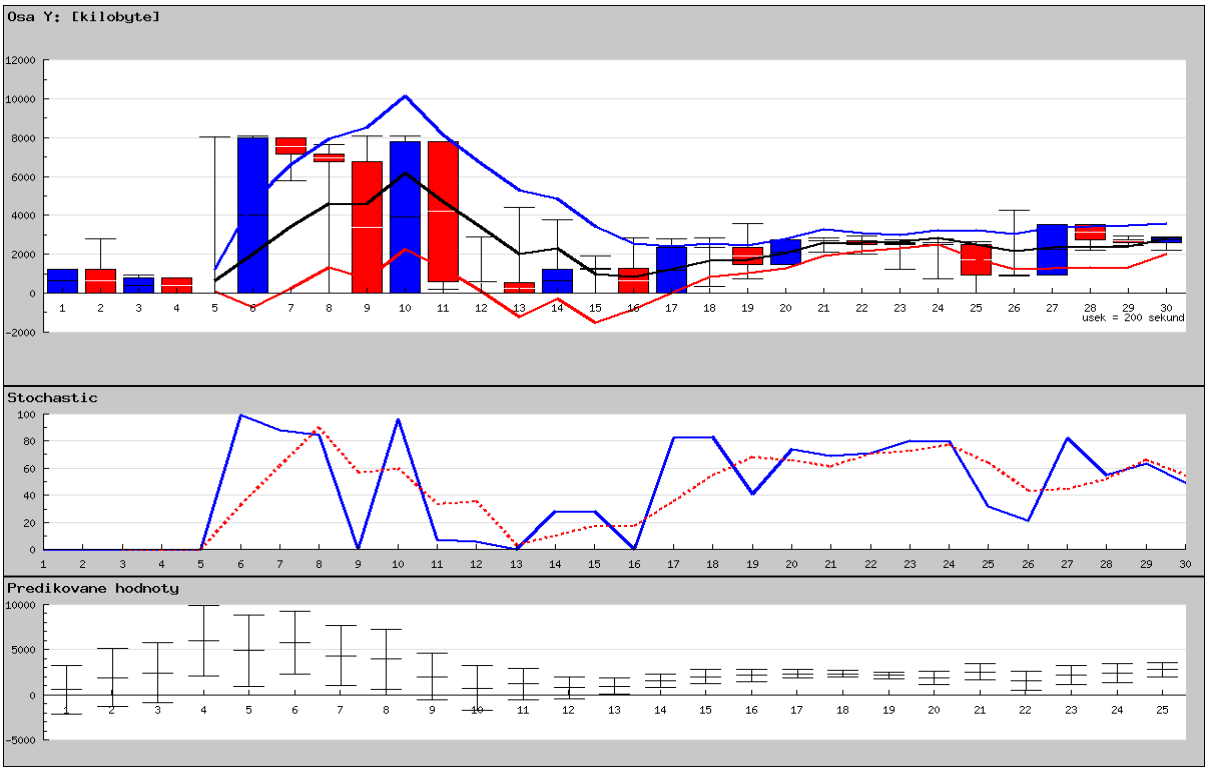
Obrázek 38: zdrojový kód - podmínky a výpočty pro získání predikce

```
/* Vygenerovani grafu pro predikci hodnot - pokud je pozadovan Bollinger
indikator */
      if ($_GET["boll"] == "on" && is_numeric($_GET["okno"])) {
            $predict = new Graph($graph_width, (int)($graph_height / 2),
"auto");
            $predict->SetScale("textlin");
            $predict->title->Set("Predikovane hodnoty");
            $predict->title->SetAlign("left");

            $predictData = NETMON_gen_predict_data($krData, $stochData,
$bollData, $_GET["perioda_sekundy"], $_GET["pocet_hodnot"], $_GET["okno"],
$_GET["kategorie_grafu"], $_GET["id"], $_GET["stoch"]);
            $predictPlot = new BoxPlot($predictData);
            $predictPlot->SetWidth(30);
            $predictPlot->SetCenter();
            $predict->Add($predictPlot);
      }
```

Obrázek 39: zdrojový kód pro vygenerování grafu predikce

Poslední uvedený úryvek pochází ze souboru *graf2.php* a popisuje využití uvedené funkce *netmon_gen_predict_data* pro vygenerování predikovaných hodnot. Je zde nastavený formát křivky a také rozměr grafu.

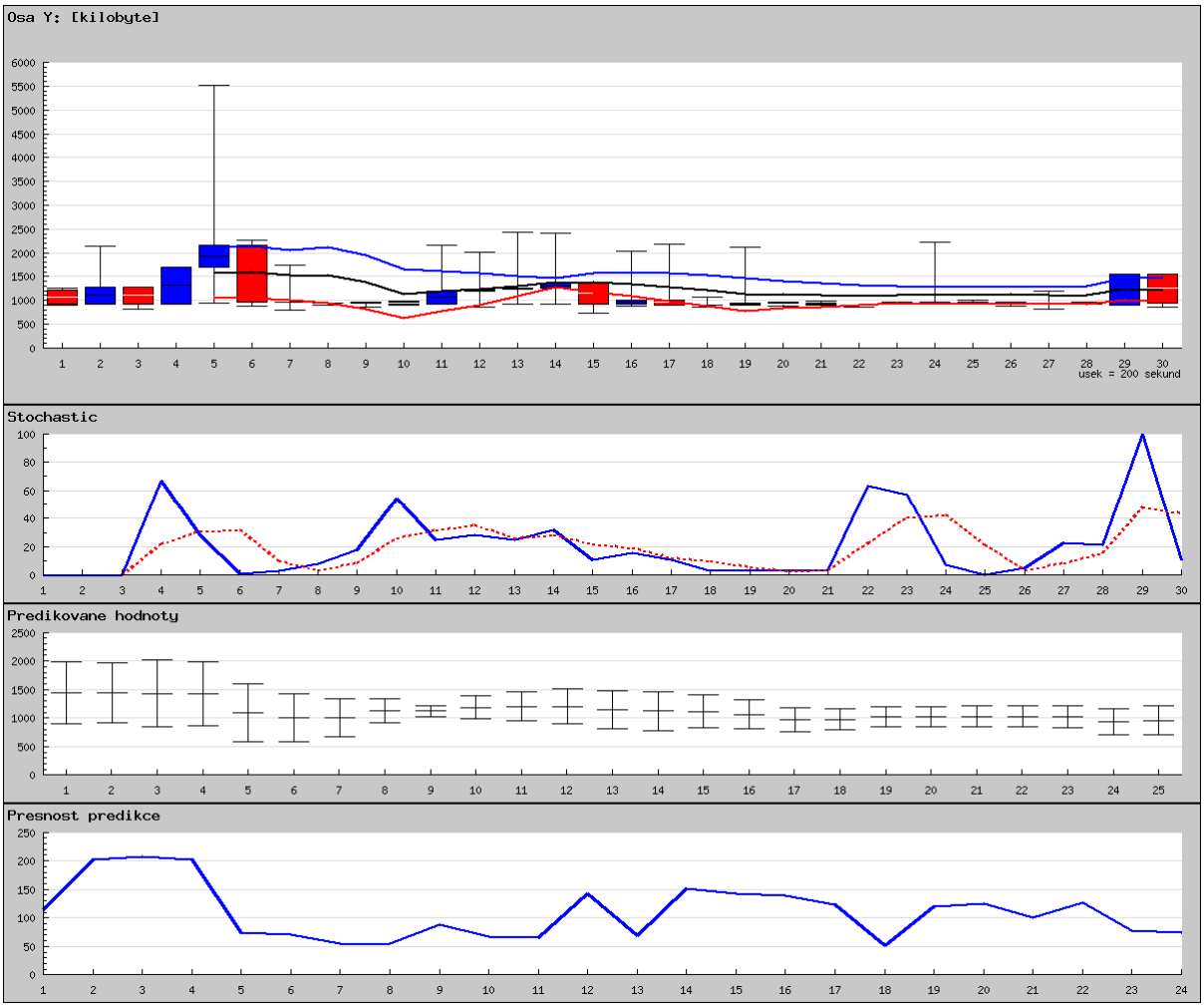
Na následujícím grafu je vidět chování predikce a lze ji zároveň porovnat s reálnými svícemi. Jak je vidět ze subjektivního pohledu predikce je velmi přesná. Predikovaná svíce v grafu je poslední svíce. Předposlední svíci v grafu predikce je možné srovnávat s poslední svící v krabicovém grafu a subjektivně posoudit kvalitu predikce.



Obrázek 40: grafický výstup predikce

Abychom mohli objektivně porovnat predikci s reálnou hodnotou, byl vytvořen graf, který uživateli umožňuje sledovat relativní přesnost predikce v průběhu měření. Jedná se o

jednoduchou funkci, která provádí podíl předikovaného maxima a reálného maxima. Minimum nyní neuvažujeme, jelikož může nabývat záporných hodnot, z důvodu odčítání odchylky ve vzorci pro Bollinger Bands. Průměrná hodnota v tuto chvíli rovněž nehraje roli, protože u krabicového grafu jsme průměrnou hodnotu získávali jednoduše součtem vstupní a výstupní hodnoty, který jsme následně vydělili dvěma za účelem korektního vykreslení grafu. Na následujícím obrázku je vidět graf přesnosti predikce. Jako největší přesnost uvažujeme hodnotu 100%. V případě že je predikce například 200%, znamená to, že reálná hodnota maxima byla rovna polovině predikované hodnoty.



Obrázek 41: porovnání predikce a reálného provozu – graf relativní přesnosti

Predikce v tomto případě slouží jen jako pomůcka uživatele sloužící k odhadnutí chování sítě v následujícím intervalu měření. Jak vypovídá graf přesnosti predikce nelze se na tuto funkci zcela spolehnout. Se zvětšujícím se intervalem měření se tato predikce pomalu ale jistě stává bezpředmětnou. Jako objektivní ji do jisté míry lze považovat jen v tich nejkratších intervalech měření. Jako důmyslnější metoda predikce by v aplikaci našla uplatnění metoda založená na umělé inteligenci. To by však představovalo hlubší prozkoumání teoretických a praktických možností využití tohoto oboru, což by mohlo být tématem jiné diplomové nebo bakalářské práce.

O vygenerování grafu přesnosti predikce se stará kód uvedený v následující ukázce.

Úryvek kódu pochází ze souboru *graf2.php* a ukazuje využití funkce *netmon_gen_accu_data*, které se předají jako parametry data krabicového diagramu (pole *$krData*) a data grafu predikce *($predictData*). Následně je z vygenerovaného pole *($accuData*) vytvořen graf znázorňující přesnost.

```
/* Vytvoreni grafu s prenosti */
                $accuData = NETMON_gen_accu_data($krData, $predictData);
                $accuGraph = new Graph($graph_width, (int)($graph_height /
2),"auto");
                $accuGraph->SetScale("textlin");
                $lineplot = new LinePlot($accuData);
                $lineplot->SetColor("blue");
                $lineplot->SetWeight(2);
                $accuGraph->Add($lineplot);
                $accuGraph->title->Set("Presnost predikce");
                $accuGraph->title->SetAlign("left")
```

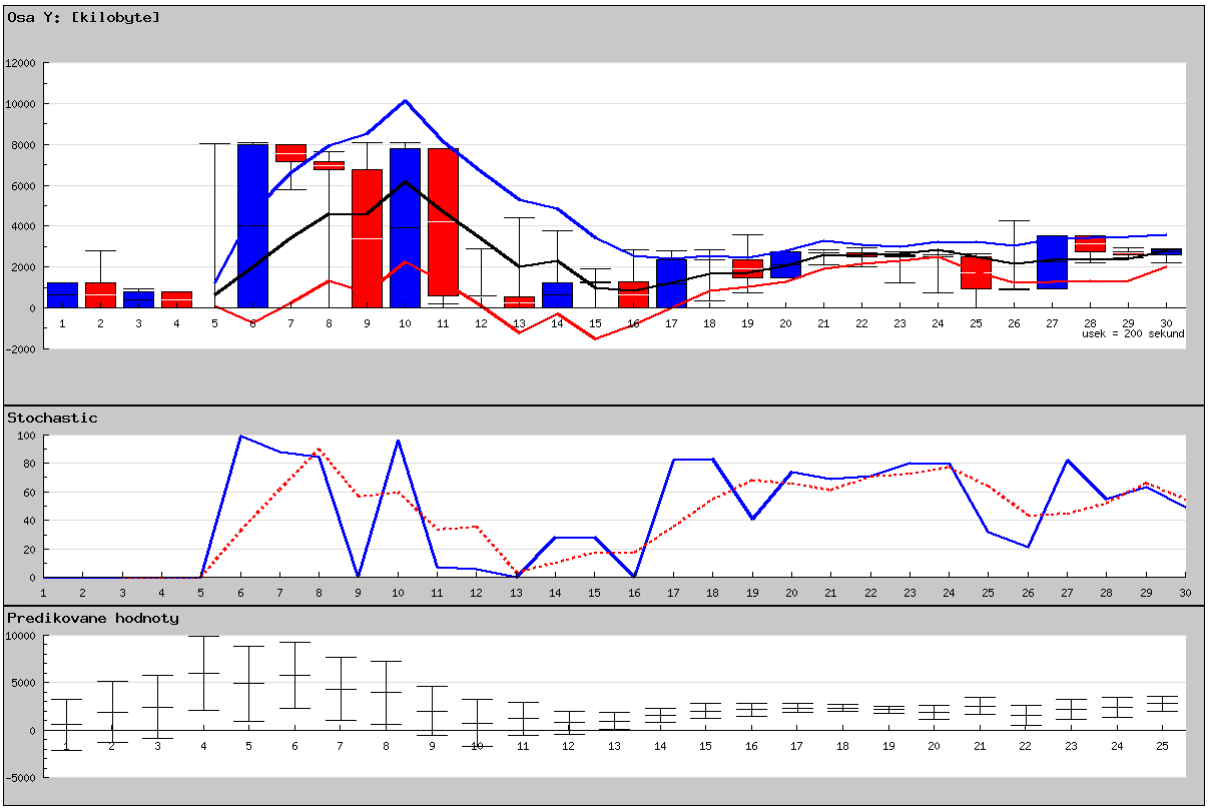
Obrázek 42: **zdrojový kód pro generování grafu přesnosti predikce**

Následující úryvek zdrojového kódu je ze souboru *design.php*. Jedná se o funkci *netmon_gen_accu_data*, která vrací pole, ze kterého je generována křivka přesnosti predikce v procentech. Tato data jsou získána z parametru, což jsou data krabicového diagramu (*$krData*) a grafu predikce *($predictData*).

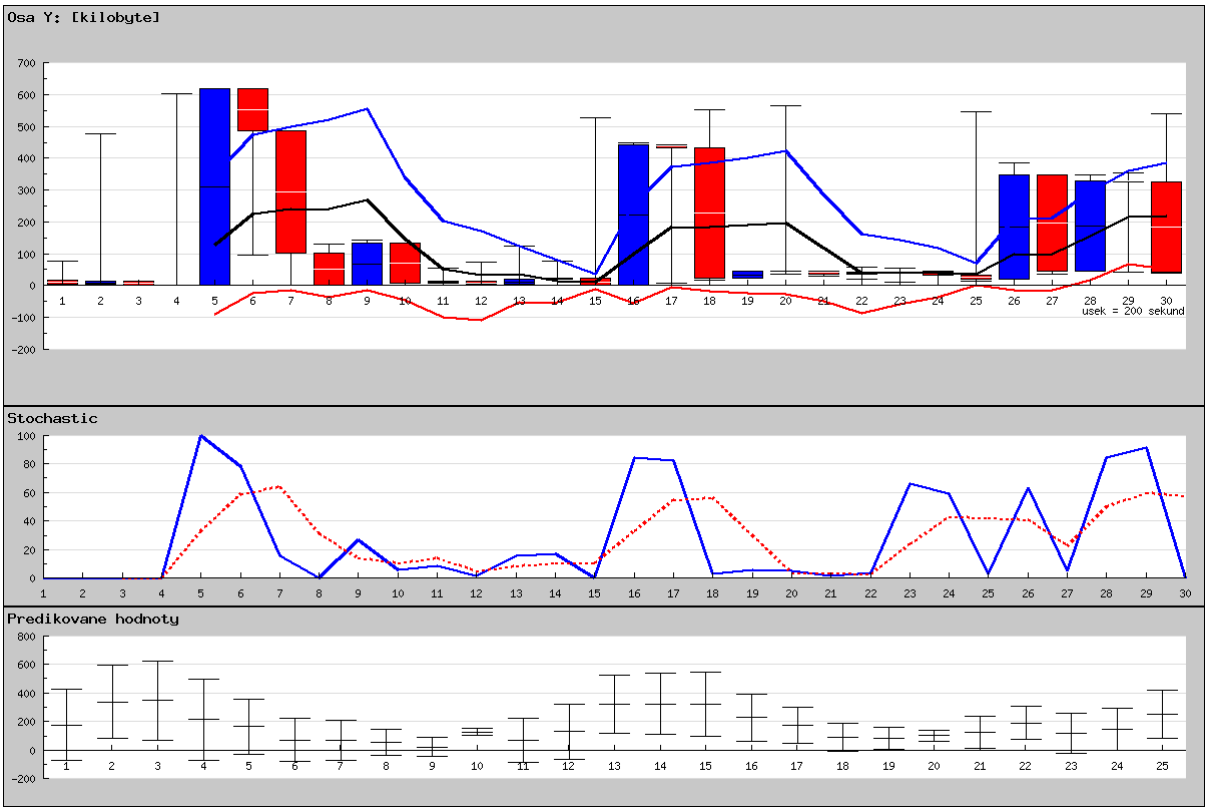
```
/* Vygeneruje data pro graf presnosti */
        function NETMON_gen_accu_data($krData, $predictData)
        {
        /* Extrahovani maxim z dat krabicoveho diagramu */
                $maxDataReal = array();
                for ($i = 0; $i < count($krData); $i++) {
                        $openVal = $krData[$i]; $i++;
                        $closeVal = $krData[$i]; $i++;
                        $minVal = $krData[$i]; $i++;
                        $maxVal = $krData[$i]; $i++;
                        $avgVal = $krData[$i];
                        $maxDataReal[] = $maxVal;
                }
        /* Extrahovani maxim z dat predikce */
                $maxDataPred = array();
                for ($i = 0; $i < count($predictData); $i++) {
                        $openVal = $predictData[$i]; $i++;
                        $closeVal = $predictData[$i]; $i++;
                        $minVal = $predictData[$i]; $i++;
                        $maxVal = $predictData[$i]; $i++;
                        $avgVal = $predictData[$i];
                        $maxDataPred[] = $maxVal;
                }
        /* Vygenerovani dat pro krivku presnosti */
                $res = array();
                for ($i = 0; $i < 24; $i++) { /* $i - index v poli
predikovanych maxim */
/* Vysledek pro krivku presnosti vyjadruje pocet procent, ktere tvori
predikovana hodnota z celku, coz je hodnota, ktera byla zjistena merenim ze
zarizeni */
       $res[] = (int)($maxDataPred[$i] / ($maxDataReal[$i + 6] / 100));
                }
```

Obrázek 43: zdrojový kód zajišťující získání dat z databáze pro graf přesnosti predikce

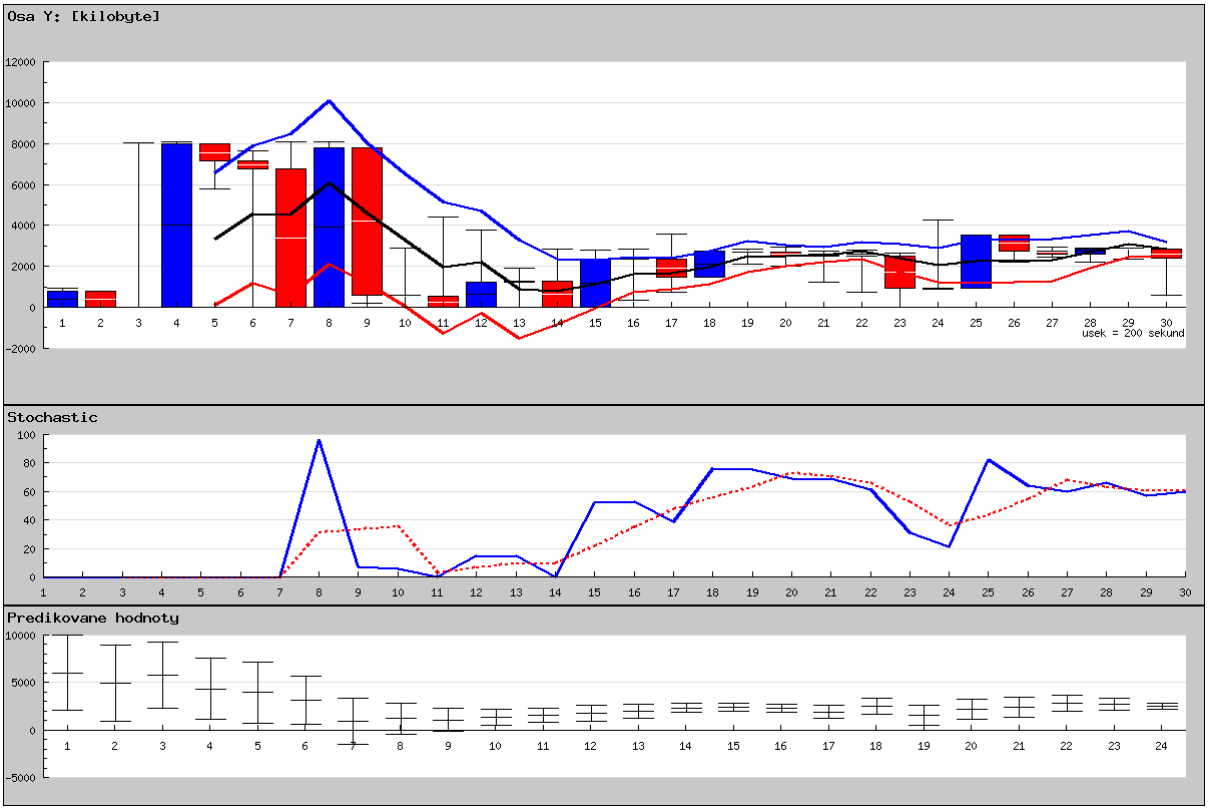
6 Testy a měření



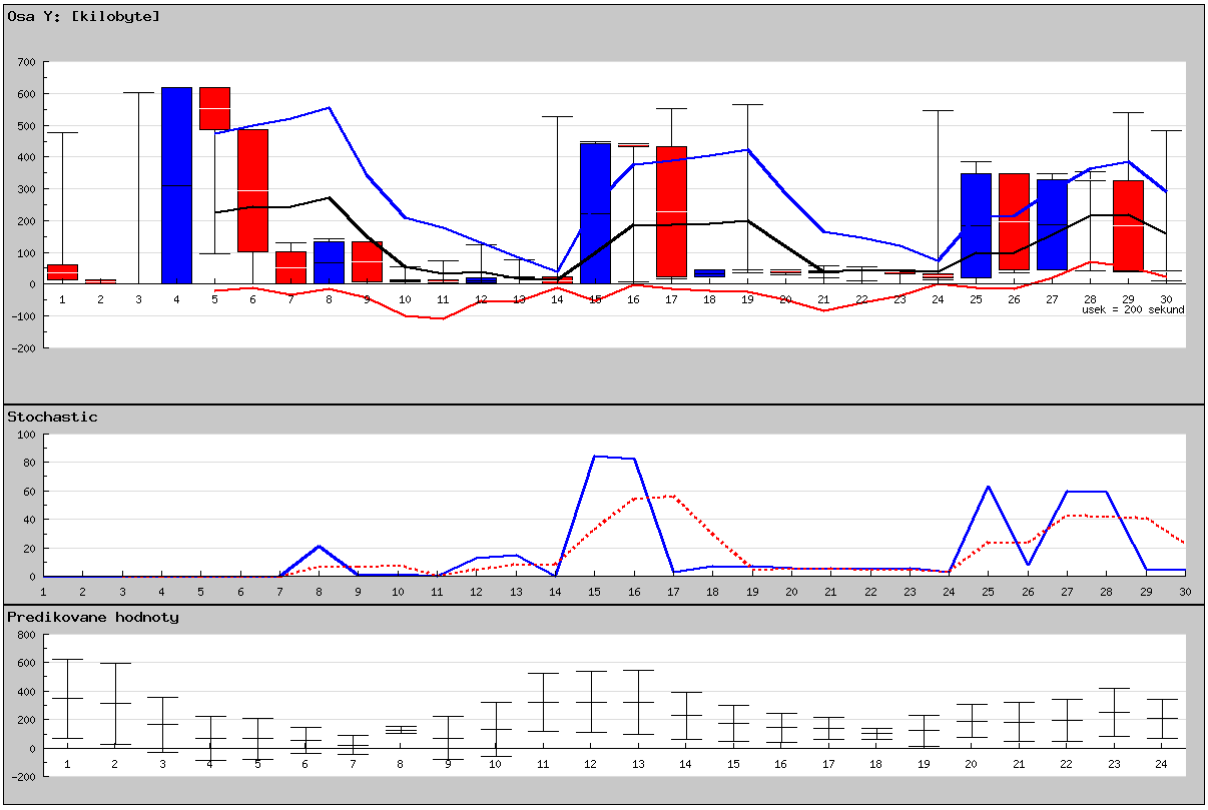
Obrázek 44: test monitorování reálného provozu – perioda měření 10, okno Stochastic 3 – DL



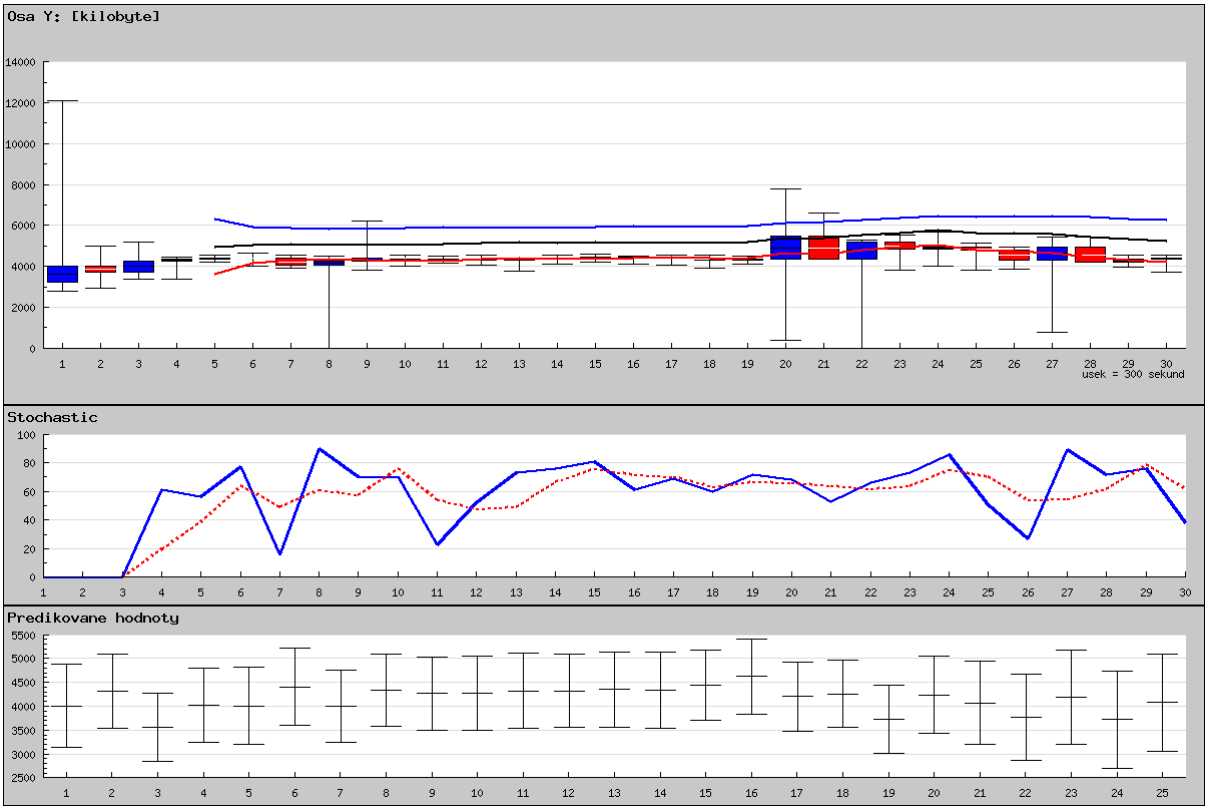
Obrázek 45: test monitorování reálného provozu – perioda měření 10, okno Stochastic 3 – UP



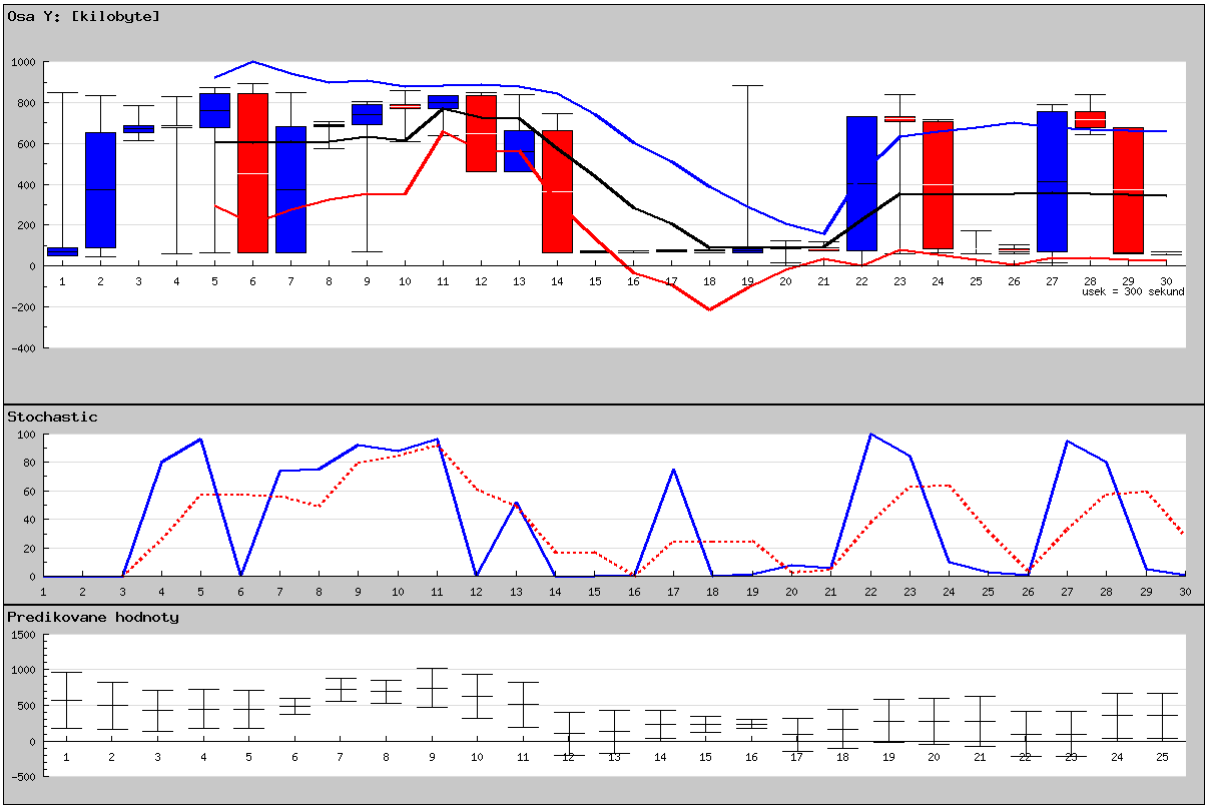
Obrázek 46: test monitorování reálného provozu – perioda měření 10, okno Stochastic 6 – DL



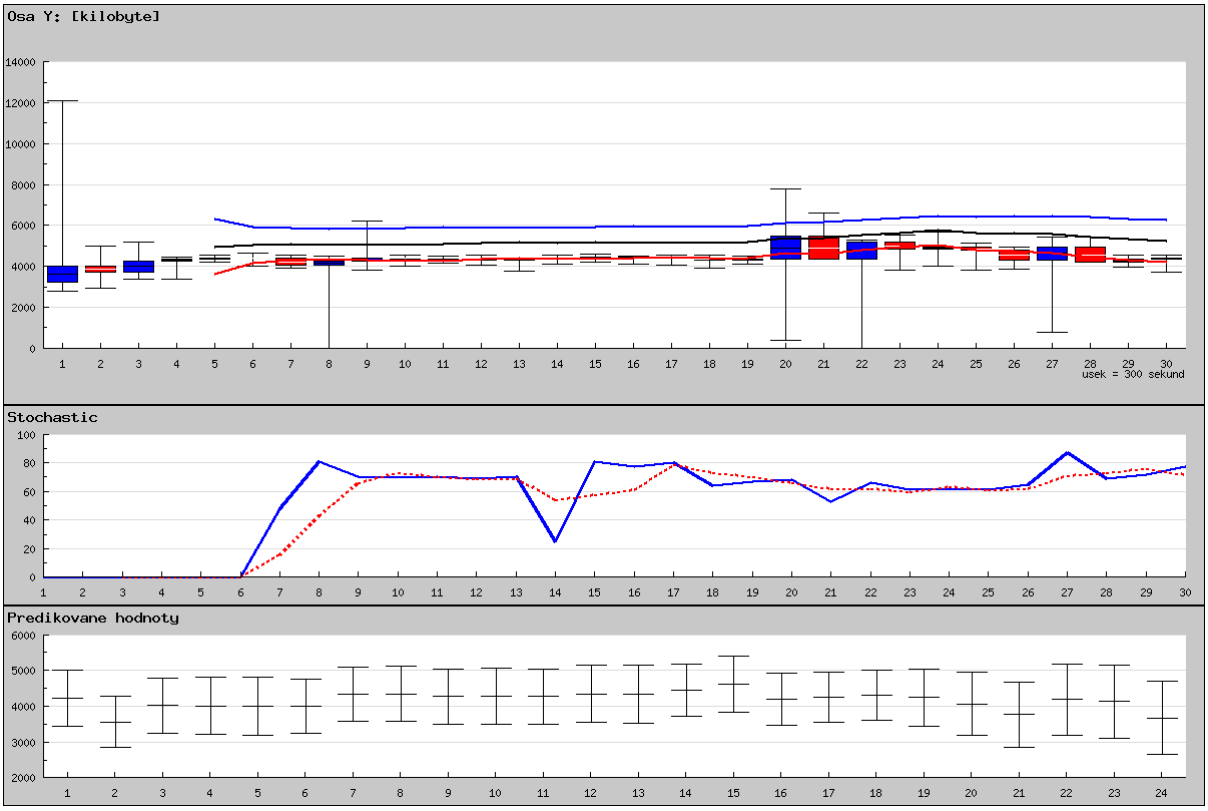
Obrázek 47: test monitorování reálného provozu – perioda měření 10, okno Stochastic 6 – UP



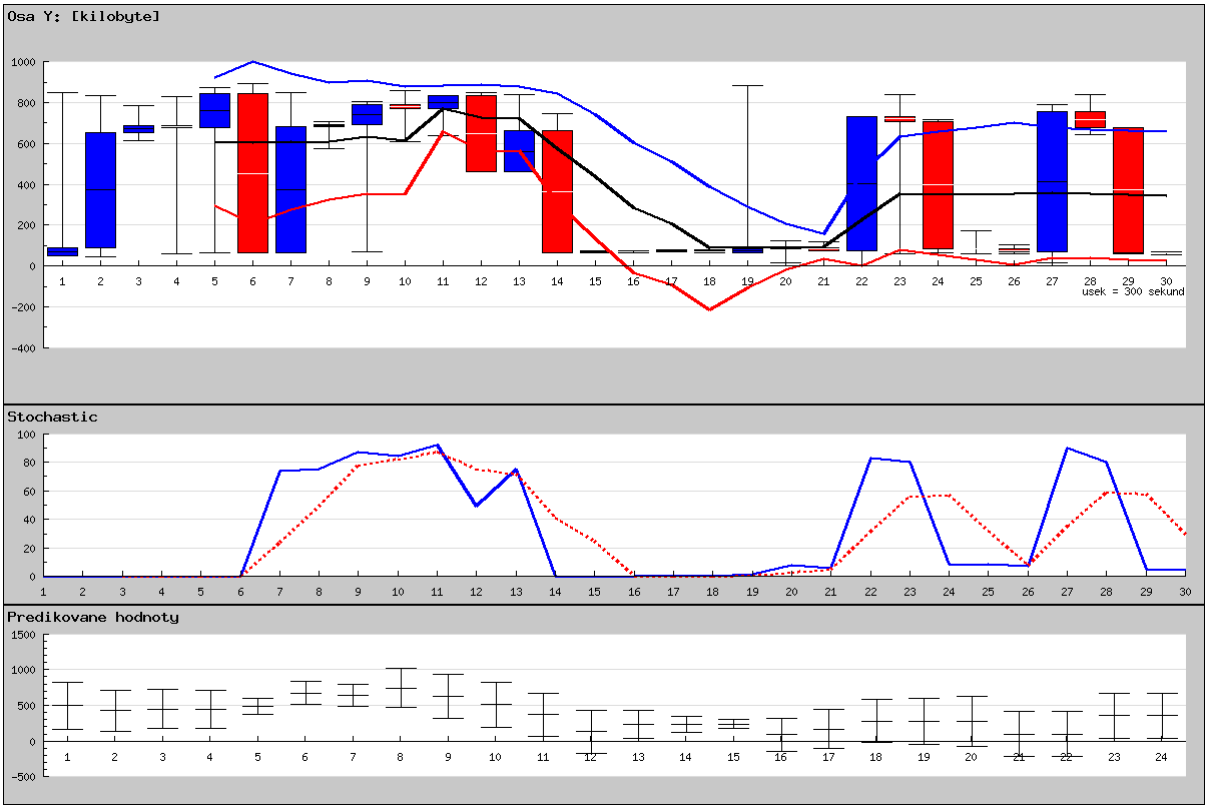
Obrázek 48: test monitorování reálného provozu – perioda měření 15, okno Stochastic 3 – DL



Obrázek 49: test monitorování reálného provozu – perioda měření 15, okno Stochastic 3 – UP



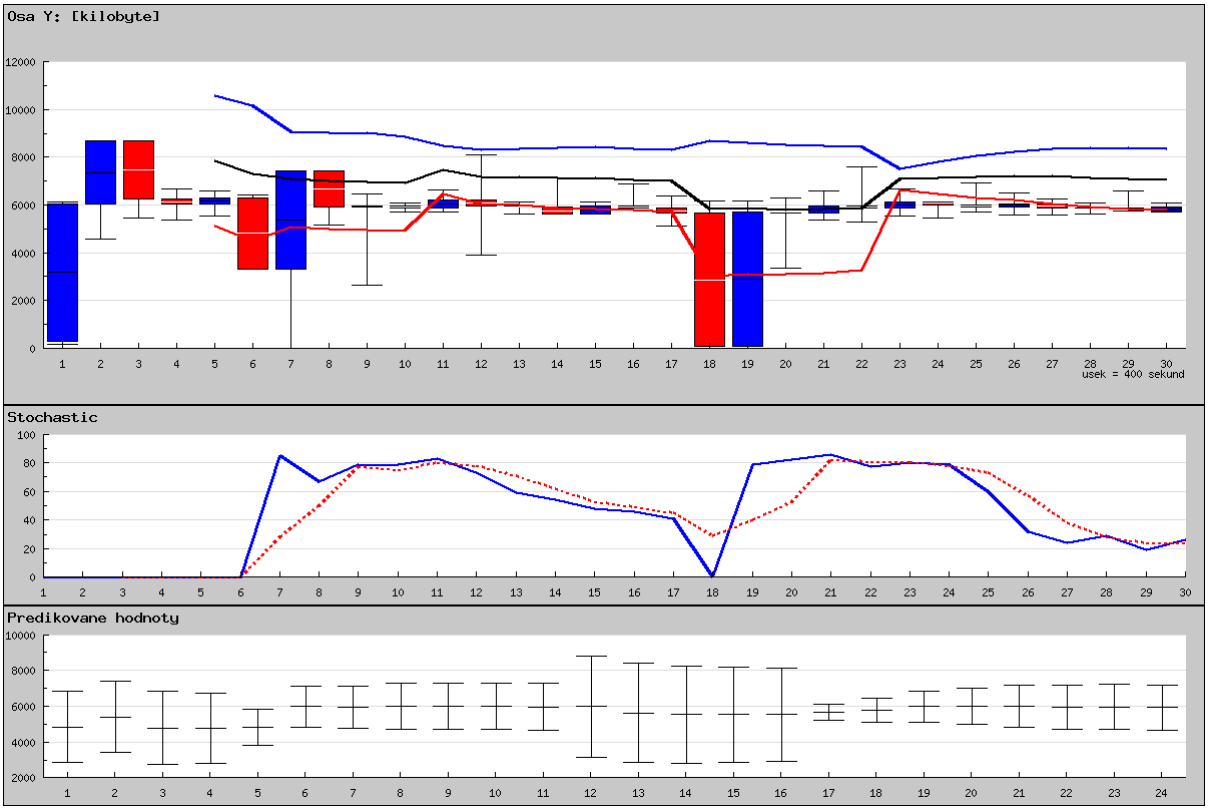
Obrázek 50: test monitorování reálného provozu – perioda měření 15, okno Stochastic 6 – DL



Obrázek 51: test monitorování reálného provozu – perioda měření 15, okno Stochastic 6 – UP

Obrázek 52: test monitorování reálného provozu – perioda měření 20, okno Stochastic 3 – DL

Obrázek 53: test monitorování reálného provozu – perioda měření 20, okno Stochastic 3 – UP



Obrázek 54: test monitorování reálného provozu – perioda měření 20, okno Stochastic 6 – DL

Obrázek 55: test monitorování reálného provozu – perioda měření 20, okno Stochastic 6 – UP

ZÁVĚR

V této diplomové práci se čtenář mohl seznámit s problematikou shromažďování a vyhodnocování informací o provozech v bezdrátových i metalických sítí. Práce se teoreticky i prakticky snaží přiblížit oblast, o které laik většinou nemá dostatečné informace. Proto zde byly zmíněny ty nejpodstatnější témata o získávání informací o stavech a přenosech v rámci sítě. Z praktického pohledu na věc existuje dnes mnoho způsobů jak ovládat sběr informací v síti. K vývoji jednoduchého informačního systému popsaného v praktické části tohoto projektu byly nutné znalosti o protokolu SNMP, platformě Mikrotik a částečně i o aplikačním rozhraní API. Platforma Mikrotik v podobě zařízení RouterBoard 433 spolu s operačním systémem RouterOS byla nakonec zvolena jako nejvhodnější varianta pro vývoj aplikace, jelikož nabízí otevřené aplikační programové rozhraní pro řízení interních funkcí přístupového bodu a podporuje protokol SNMP. Fakt, že zařízení není založeno na volně šiřitelném softwarovém řešení, nebrání vývoji podobných aplikací, jelikož zařízení disponuje širokým spektrem funkcí, které lze pomocí API a protokolu SNMP konfigurovat a využívat. V dosáhnutém stavu aplikace může být tento systém nasazen do reálné sítě o libovolném počtu zařízení Mikrotik a libovolném počtu klientských zařízení. Celá aplikace byla konstruována tak, aby byla otevřená pro jakékoliv další úpravy a vylepšení.

SEZNAM ZKRATEK

ABI (*Application Binary Interface*)
API (*Aplication Programming Interface*)
BER (*Basic Encoding Rules*)
DMI (*Desktop Management Interface*)
DNS (*Domain Name Server*)
EGP (*Exterior Gateway Protocol*)
GPL (*Genreal Public Licence*)
ICMP (*Internet Control Message Protocol*)
IP (*Internet Protocol*)
JDK (*Java Developement Kit*)
MFI (*Management Information Format*)
MIB (*Management Information Base*)
MVC (*Model-View-Controller*)
NAT (*Network Address Translation*)
OID (*Object Identifier*)
OSPF(*Open Shortest Path Firs*t)
PPP (*Point-to-Point Protocol*)
QoS (Quality of Service)
RFC (*Request for Comments*)
RIP (*Routing Information Protocol*)
RMON (*Remote Network Monitoring*)
SNMP (*Simple Network Management Protocol*)
SSH (*Secured Shell*)
TCP (*Transmission Kontrol Protocol)*
UDP (*User Datagram Protocol*)
UPnP (*Univeresal Plug and Play*)
WPA2(*Wifi Protected Access*)

REFERENCE

[1] J. BIGELOW, Stephen. *Mistrovství v počítačových sítích : Správa konfigurace, diagnostika a řešení prbolémů*. Brno : Computer Press, 2004. 992 s. ISBN 80-251-0178-9.

[2] Application programming interface. In *Wikipedia : the free encyclopedia* [online]. St. Petersburg (Florida) : Wikipedia Foundation, 30 July 2001, last modified on 4 December 2010 [cit. 2010- 12-06]. Dostupné z WWW : <http://en.wikipedia.org/wiki/Application_ programming_interface>.

[3] OpenWrt. In *Wikipedia : the free encyclopedia* [online]. St. Petersburg (Florida) :Wikipedia Foundation, 5 May 2005, last modified on 5 December 2010 [cit. 2010-12 06]. Dostupné z WWW: <http://en.wikipedia.org/wiki/OpenWrt>.

[4] ŠTRAUCH, Adam. *Root.cz* [online]. 7.11.2008 [cit. 2010-12-06]. Mikrotik: seznámení s Wi-Fi krabičkou. Dostupné z WWW: <http://www.root.cz/clanky/mikrotik-seznameni-s-wi-fi-krabickou/>.

[5] *MikroTik Wiki* [online]. 2008, 06.12.2010 [cit. 2010-12-06]. Dostupné z WWW: <http://wiki.mikrotik.com/wiki/Main_Page>.

[6] DD-WRT. In *Wikipedia : the free encyclopedia* [online]. St. Petersburg (Florida) : Wikipedia Foundation, 19 November 2005, last modified on 12 December 2010 [cit. 2010-12-16]. Dostupné z WWW: <http://en.wikipedia.org/wiki/DD-WRT>.

[7] ELDER Dr., Alexander. *Trading for living*. New York : [s.n.], 1992. 179 s.