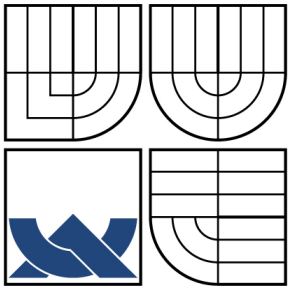
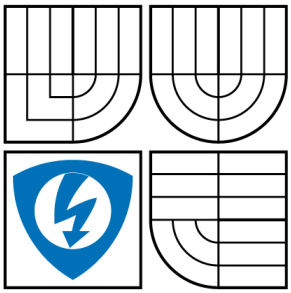


VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ
ÚSTAV TELEKOMUNIKACÍ

FACULTY OF ELECTRICAL ENGINEERING AND COMMUNICATION
DEPARTMENT OF TELECOMMUNICATIONS

APLIKACE PRO MONITOROVÁNÍ A SPRÁVU SÍTĚ PRO ISP

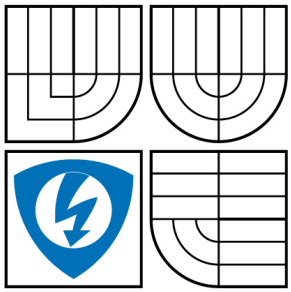
APPLICATION FOR NETWORK MONITORING AND MANAGEMENT FOR ISP

DIPLOMOVÁ PRÁCE
MASTER'S THESIS

AUTOR PRÁCE — Bc. TOMÁŠ KMONÍČEK
AUTHOR

VEDOUCÍ PRÁCE — Ing. MILAN ŠIMEK
SUPERVISOR

BRNO 2008



VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

Fakulta elektrotechniky a komunikačních technologií

Ústav telekomunikací

Diplomová práce

magisterský navazující studijní obor
Telekomunikační a informační technika

Student: Kmoníček Tomáš Bc. ***ID:*** 89846
Ročník: 2 ***Akademický rok:*** 2007/2008

NÁZEV TÉMATU:

Aplikace pro monitorování a správu sítě pro ISP

POKYNY PRO VYPRACOVÁNÍ:

Prostudujte možnosti monitorování a správy uživatelů, aktivních prvků a služeb rozsáhlejší počítačové sítě. Dle nabytých znalostí navrhněte sytém pro správu zákazníků, monitorování provozu a funkčnosti sítě a jejích služeb. Navrženou aplikaci otestujte při reálném provozu ISP sítě.

DOPORUČENÁ LITERATURA:

[1] S.J. Bigelow, Mistrovství v počítačových sítích, ComputerPress, a.s, 2004, ISBN: 80-251-0178-9
[2] R. PUŽMANOVÁ, Moderní komunikační sítě od A d Z, Computer Press, a.s., 2005, ISBN: 80-251-1278-0

Termín zadání: 11.2.2008 ***Termín odevzdání:*** 28.5.2008

Vedoucí práce: Ing. Milan Šimek

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

LICENČNÍ SMLOUVA
POSKYTOVANÁ K VÝKONU PRÁVA UŽÍT ŠKOLNÍ DÍLO

uzavřená mezi smluvními stranami:

1. Pan/paní

Jméno a příjmení: Bc. Tomáš Kmoníček
Bytem: Harrachov 241, 51246, Harrachov
Narozen/a (datum a místo): 31.5.1984, Jilemnice

(dále jen "autor")

a

2. Vysoké učení technické v Brně

Fakulta elektrotechniky a komunikačních technologií
se sídlem Údolní 244/53, 60200 Brno 2
jejímž jménem jedná na základě písemného pověření děkanem fakulty:
prof. Ing. Kamil Vrba, CSc.

(dále jen "nabyvatel")

Článek 1
Specifikace školního díla

1. Předmětem této smlouvy je vysokoškolská kvalifikační práce (VŠKP):
   - ☐ disertační práce
   - ☑ diplomová práce
   - ☐ bakalářská práce

   jiná práce, jejíž druh je specifikován jako .....

(dále jen VŠKP nebo dílo)

Název VŠKP: Aplikace pro monitorování a správu sítě pro ISP
Vedoucí/školitel VŠKP: Ing. Milan Šimek
Ústav: Ústav telekomunikací
Datum obhajoby VŠKP: .....

VŠKP odevzdal autor nabyvateli v:

- ☑ tištěné formě - počet exemplářů 1
- ☑ elektronické formě - počet exemplářů 1

2. Autor prohlašuje, že vytvořil samostatnou vlastní tvůrčí činností dílo shora popsané a specifikované. Autor dále prohlašuje, že při zpracovávání díla se sám nedostal do rozporu s autorským zákonem a předpisy souvisejícími a že je dílo dílem původním.
3. Dílo je chráněno jako dílo dle autorského zákona v platném znění.
4. Autor potvrzuje, že listinná a elektronická verze díla je identická.

Článek 2

Udělení licenčního oprávnění

1. Autor touto smlouvou poskytuje nabyvateli oprávnění (licenci) k výkonu práva uvedené dílo nevýdělečně užít, archivovat a zpřístupnit ke studijním, výukovým a výzkumným účelům včetně pořizovaní výpisů, opisů a rozmnoženin.
2. Licence je poskytována celosvětově, pro celou dobu trvání autorských a majetkových práv k dílu.
3. Autor souhlasí se zveřejněním díla v databázi přístupné v mezinárodní síti
   - ☑ ihned po uzavření této smlouvy
   - ☐ 1 rok po uzavření této smlouvy
   - ☐ 3 roky po uzavření této smlouvy
   - ☐ 5 let po uzavření této smlouvy
   - ☐ 10 let po uzavření této smlouvy

   (z důvodu utajení v něm obsažených informací)
4. Nevýdělečné zveřejňování díla nabyvatelem v souladu s ustanovením § 47b zákona č. 111/1998 Sb., v platném znění, nevyžaduje licenci a nabyvatel je k němu povinen a oprávněn ze zákona.

Článek 3

Závěrečná ustanovení

1. Smlouva je sepsána ve třech vyhotoveních s platností originálu, přičemž po jednom vyhotovení obdrží autor a nabyvatel, další vyhotovení je vloženo do VŠKP.
2. Vztahy mezi smluvními stranami vzniklé a neupravené touto smlouvou se řídí autorským zákonem, občanským zákoníkem, vysokoškolským zákonem, zákonem o archivnictví, v platném znění a popř. dalšími právními předpisy.
3. Licenční smlouva byla uzavřena na základě svobodné a pravé vůle smluvních stran, s plným porozuměním jejímu textu i důsledkům, nikoliv v tísni a za nápadně nevýhodných podmínek.
4. Licenční smlouva nabývá platnosti a účinnosti dnem jejího podpisu oběma smluvními stranami.

V Brně dne: .....

....

Nabyvatel Autor

ABSTRAKT

Diplomová práce se zaobírá návrhem dohledového a administrativního systému pro poskytovatele Internetu. Výsledkem práce je plně funkční dohledový systém umožňující správci sítě kontrolu odezvy a ztrátovosti v celé síti. Dále zjišťování a zaznamenávání dat získávaných pomocí protokolu SNMP ze síťových prvků, jako například množství přenesených dat prvkem aj. Vše je možné zobrazit ať už v souhrnných přehledech či v grafech za období až jednoho roku. Druhou administrativní částí systému je plnohodnotná klientská databáze. Zde je možné spravovat klienty, jim poskytované služby, platby, přípojky a IP adresy. Pro účely kontrolování plateb je systémem nabízen import výpisů z banky. Ke klientské databázi i dohledové aplikaci je přistupováno pomocí zabezpečené webové aplikace, aby nebylo nutné pořizování jakéhokoli nadbytečného software. Programová část systému je realizována v Pythonu a webová aplikace pomocí PHP s využitím AJAX, pro přiblížení webové aplikace běžným počítačovým aplikacím. Systém je navíc navržen pro tzv. přídavné moduly, které umožňují rozšíření systému o takřka jakékoli další funkce.

KLÍČOVÁ SLOVA

aplikace dohled administrace klient ISP síť databáze SNMP ping ztrátovost paketů

ABSTRACT

The objective of this master's thesis is to develop an ISP application for service monitoring and management. A fully functioning monitoring software has been developed, which allows the administrator to check the round-trip time and the packet loss in the entire network. Another function of the application is obtaining information from network devices by the SNMP protocol (e.g. packet and byte count etc.). The software uses the information to create hour, day, week or year graphs. All graphs and statistics are visualised at the administration webpages. For the purpose of client administration, a clients database has been developed. This part of the application contains functions of payment administration, IP addresses, contacts etc. Both parts of the web pages have been developed using PHP and AJAX, which makes the application similar to standard desktop applications. The testing application has been developed using Python and the entire application has been designed for modules, which can be later used for the development of other functions.

KEYWORDS

aplication monitoring management client ISP network database SNMP ping packet lost

PROHLÁŠENÍ

Prohlašuji, že svou diplomovou práci na téma „Aplikace pro monitorování a správu sítě pro ISP“ jsem vypracoval samostatně pod vedením vedoucího diplomové práce 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 této diplomové práce 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 ....

(podpis autora)

PODĚKOVÁNÍ

Především bych chtěl poděkovat vedoucímu práce, kterým byl pan Ing. Milan Šimek, za jeho ochotu a rady při vypracování diplomové práce.

Bc. Tomáš Kmoníček

OBSAH

Úvod **12**

1 Seznámení s problematikou **13**
1.1 Typ sítě ISP . . . 13
1.2 Síťové prvky a jejich možnosti . . . 13
1.2.1 SNMP . . . 14
1.3 Parametry sítě . . . 15
1.4 Ostatní prostředky . . . 16

2 Programovací jazyk **17**
2.1 Python . . . 17
2.2 PHP . . . 18

3 Realizace aplikace **20**
3.1 Testovací aplikace NetScan . . . 20
3.1.1 Hlavní program . . . 22
3.1.2 Spouštěč . . . 23
3.1.3 Správce databáze . . . 25
3.1.4 Správce zpráv . . . 27
3.1.5 ICMP test . . . 28
3.1.6 Port test . . . 30
3.1.7 Struktura modulu . . . 30
3.1.8 Modul SNMP . . . 31
3.1.9 Modul TCPd . . . 33
3.2 Webové aplikace systému . . . 33
3.2.1 Dohledová aplikace . . . 34
3.2.2 Klientská databáze . . . 39
3.2.3 Zabezpečení . . . 41
3.3 Databáze RRD . . . 41
3.4 Databáze PostgreSQL . . . 44
3.4.1 Blok služeb a plateb . . . 46
3.4.2 Blok kontaktů . . . 46
3.4.3 Blok přípojek . . . 48
3.4.4 Blok objektů . . . 49
3.4.5 Blok síťových prvků . . . 50

4 Instalace a nastavení aplikace **53**
4.1 Instalace 53
4.2 Nastavení NetScan.conf 54

5 Nasazení a testy aplikace **55**

6 Závěr **57**

Literatura **58**

Seznam symbolů, veličin a zkratek **59**

Seznam příloh **61**

A Schéma navržené SQL databáze **62**

B Webové stránky aplikace **63**
B.1 Dohledová aplikace NetScan 63
B.2 Přehled síťového prvku 64

SEZNAM OBRÁZKŮ

1.1 Příklad topologie rozvětvené hvězdy 14
3.1 Blokové schéma jádra testovací aplikace 20
3.2 Diagram spuštění aplikace 21
3.3 Vývojový diagram správce databáze 26
3.4 Blokové schéma funkce bloku ping 29
3.5 Blokové schéma modulu SNMP 33
3.6 Ukázka dynamického stromu 35
3.7 Vývojový diagram obnovy dynamického stromu 35
3.8 Příklad grafu průběhu odezvy 37
3.9 Webová stránka kartotéky klientů 39
3.10 Webová stránka editace klienta 40
3.11 Princip Round-Robin databáze 42
3.12 Příklad vzniku primárních datových bodu a RRA 43
3.13 Blokové zobrazení částí databáze 44
3.14 Tabulky bloku služeb a plateb 47
3.15 Tabulky bloku kontaktů 47
3.16 Tabulky bloku síťových prvků 52
5.1 Graf vytížení CPU serveru v reálné aplikaci 56

SEZNAM TABULEK

3.1 Přehled obsluhy signálů OS Linux 23
3.2 Přehled RRA vytvářených v základním nastavení 43
3.3 Popis polí tabulky uživatelů a klientů 44
3.4 Popis bitů oprávnění 45
3.5 Popis polí tabulek bloku služeb a plateb 48
3.6 Popis polí tabulek kontaktů 49
3.7 Popis polí tabulek bloku přípojek 49
3.8 Popis polí tabulky objektů 50
3.9 Popis polí tabulek bloku síťových prvku 51
4.1 Popis možných nastavení NetScan.conf 54
5.1 Parametry testovacího serveru 55

ÚVOD

Tato diplomová práce se věnuje problematice monitorování a správy ethernetové sítě takzvaných ISP (poskytovatel internetových sluďžeb – Internet Service Provider). S dnešním velice významným rozvojem sítě Internet se objevuje mnoho malých lokálních sítí. Rozměry těchto sítí jsou od několika počítačů až po tisíce. U mnoha i rozsáhlejších sítí se velice často objevují různé obtíže spojené s přetížením linek či s použitím levnějších a méně kvalitních síťových prvků případně s nejrůznějšími útoky na tuto síť. Vzhledem k těmto skutečnostem je nutné sledovat parametry jednotlivých linek, aby byla zajištěna maximální kvalita služeb. Ze základních parametrů, které je nutné sledovat je odezva a ztrátovost paketů. Sledování těchto parametrů je možno realizovat za pomocí protokolů architektury TCP/IP (Transmission Control Protocol / Internet Protocol) jako je například protokol ICMP (Internet Control Message Protocol).

Aby bylo také možné případný problém snáze řešit, je vhodné mít k dispozici další informace o síti, jako jsou například počty spojení na daném prvku, přehled množství dat přenesených síťovým prvkem, případně podrobnosti o datovém toku či údaje o připojených počítačích. Cílem této práce je vytvořit dohledový systém, který umožní správci dané sítě přehledně sledovat parametry sítě a případně pomocí systému co nejsnáze řešit běžné problémy v síti. Dále výsledná aplikace bude umožňovat správu uživatelů (resp. zákazníků) poskytovatele. Dohledová část aplikace bude s databází klientů propojena pro zefektivnění komunikace se zákazníke v případě závad na straně zákazníka.

Vzhledem ke skutečnosti, že na dnešním trhu existuje mnoho různých síťových prvků, které nabízí menší či větší možnosti dohledu, bude aplikace navrhována tak, aby byl co možná nejméně závisela na síťových prvcích použitých v dané síti.

1 SEZNÁMENÍ S PROBLEMATIKOU

1.1 Typ sítě ISP

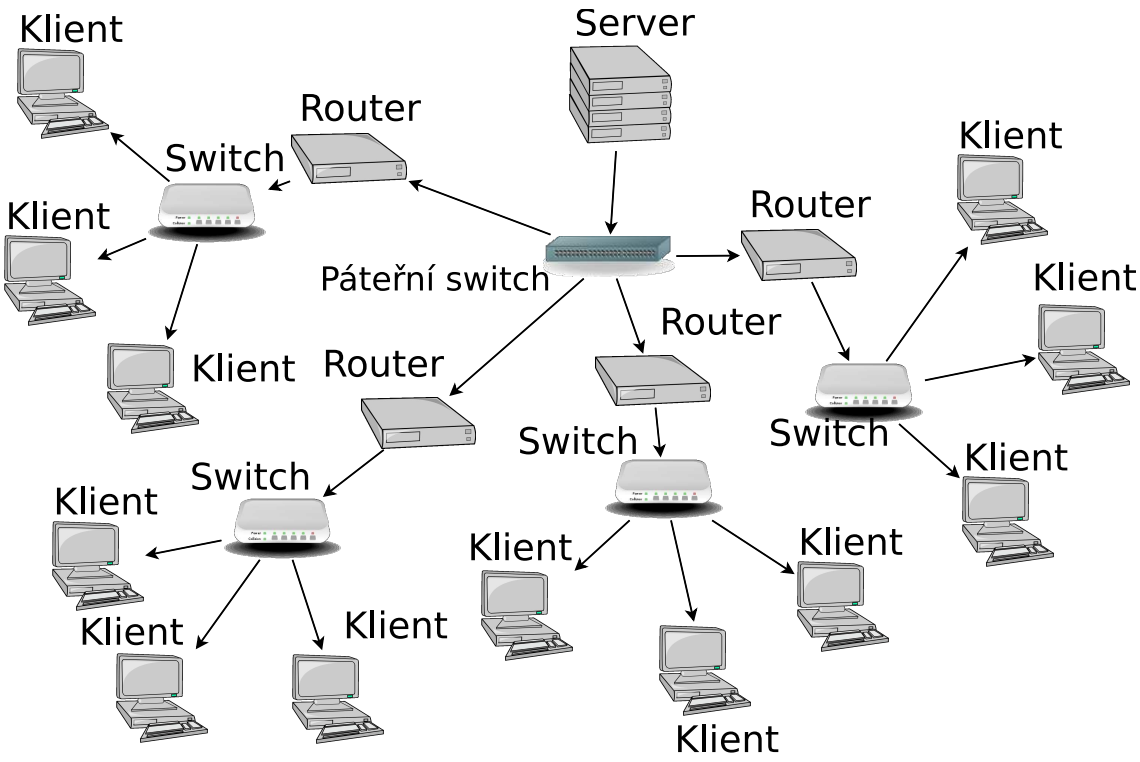
Vzhledem k již zmiňovanému velkému počtu ISP, je zřejmé, že je použito různých typů sítí. Rozdíly jsou v topologii celé sítě a především v použitých přenosových mediích, s čímž souvisí použití různých síťových prvků (viz. kapitola 1.2). Pro realizaci dohledového systému je velice podstatná topologie sítě. Pokud má být systém realizován jako aplikace provozovaná pouze na serveru, aby se co nejvíce eliminovala závislost na použitých síťových prvcích, je nutné, aby server měl co možná nejtransparentnější přistup k celé sítí. Nejčastější topologií je v dnešní době rozvětvená hvězda. Příklad takovéto sítě je na obrázku 1.1. U tohoto typu sítě je pro server transparentní přístup k prvkům sítě snadný, pokud není na nějakém z routerů provozován NAT (překlad IP adres – Network Address Translation). Pokud je použit NAT nemůže server přistupovat k prvků za bodem, kde je NAT aplikován, neboť tyto prvky jsou z hlediska adres maskována za jednou definovanou adresou. Proto je vhodné, aby byl server umístěn v bodě, z něhož bude k ostatním prvkům sítě přistopovat přímo, nikoli přes NAT.

Ve většině případů je síť poskytovatele rozdělena na menší podsítě a za pomocí směrovačů (routerů) je komunikace směrována dále sítí. Toto rozdělení navíc umožňuje snadné zálohování spojení, které je v dnešní době stále více vyžedováno a stává se dnes, obzvláště u roměrnějších sítí, takřka nezbytností.

Sítě poskytovatelů internetu jsou ve většině případů postaveny na architektuře typu TCP/IP (Transmission Control Protocol / Internet Protocol), neboť se jedná o architekturu sítě Internet a není proto nutné jiné přizpůsobení. Fyzická vrstva je velmi často realizována sítěmi typu Ethernet či bezdrátovými spoji. S bezdrátovým spojení se objevuje například možnost ztrátovosti v případě zarušení spoje či špatných povětrnostních podmínek, proto je nutné obzvláště tyto cesty v síti prověřovat zda splňují požadované parametry a je tedy monžné zákazníkovi zajistit odpovídající služby.

1.2 Síťové prvky a jejich možnosti

V sítích ISP je velmi často použito různých síťových prvků. Prvky se liší kvalitou a především funkcemi, které prvky nabízí. Koncovými prvky sítě jsou zpravidla počítače klientů, případně celé sítě těchto klientů. Z hlediska dohledu a správy sítě ISP jsou tyto koncové prvky nejméně důležité, neboť na ně není možné klást specifické



Obr. 1.1: Příklad topologie rozvětvené hvězdy

požadavky. Z pravidla však směrovací prvek pro danou podsíť vlastní IP adresu a tu je možné testovat pomocí protokolu ICMP.

Podstatnými prvky sítě jsou naopak veškeré páteřní prvky, směrovače v uzlech sítě a také switche v lokálních podsítích. Jako switche v lokálních podsítích se však často používají takové, jejichž cena je nižší, a tak nenabízí takřka žádné možnosti dohledu ani správy. V takto navržené podsíti je možné pouze provádět analýzu pomocí sledování koncových zařízení uživatelů a charakteristiky využití sítě na směrovacím prvku podsítě. Směrovače dané podsítě uchovávají tzv. ARP (protokol pro rozlišení síťových adres – Address Resolution Protocol) záznamy. Jedná se o fyzické adresy jednotlivých zařízení, počítačů, případně dalších směrovačů v podsíti, jenž jsou v současné době aktivní a komunikují tedy přes směrovač. Získáním a sledováním těchto údajů je možné například odhalovat zneužívání IP adresy jiného uživatele aj. Možnost získávat tyto adresy však závisí na možnostech směrovače.

1.2.1 SNMP

Jako páteřní a směrovací prvky jsou většinou použita zařízení, která implementují minimálně protokol SNMP (jednoduchý protokol pro správu – Simple Network Management Protocol). Tento protokol umožňuje získávat různé informace a také měnit jednotlivá nastavení daného prvku. Množství získatelných a nastavitelných parame-

trů se u prvků samozřejmě liší v závislosti na množství funkcí, která např. směrovač poskytuje. Seznam argumentů je sdružován v tzv. MIB (Management Information Base) databázi, která dovoluje jednoznačně identifikovat informace využívané systémem správy. Aby mohl SNMP manager i agent tyto informace získat a předávat, tak je nutná znalost struktury MIB. Báze dat je objektově orientovaná. Data jsou uložena jako objekty a sdružují se do tříd. Jednotlivé objekty mají určité hodnoty. Každý řízený objekt v MIB obsahuje veškeré informace potřebné pro popis. Způsob pojmenování objektů je založen na jejich vztahu. Jeden objekt může obsahovat jiné objekty nebo jiné třídy. MIB je tedy tvořena jedním stromem. Využitím SNMP protokolu lze tedy snadno získávat například informace o datovém toku daným prvkem, jeho logové informace, či již zmiňované fyzické adresy komunikujících zařízení.

1.3 Parametry sítě

Pro potřeby snadného rozboru problému v síti je nutné, aby správce měl k daným prvků potřebné údaje, z nichž je možné případnou poruchu odhalit. U každého prvku určitého typu je však nutné sledovat jiné parametry. Některé parametry se dají sledovat jednoduše, neboť je možné je získávat pomocí již zmiňovaného protokolu SNMP, jiné jako například časová odezva či ztrátovost trasy je nutno měřit a obnovovat.

Z tohoto vyplývá, že dohledový systém musí realizovat opakované testování časové odezvy jednotlivých prvků sítě. Tato měření se realizují snadno pomocí protokolu ICMP (Internet Control Message Protocol), což je jeden z jádrových protokolů ze sady protokolů Internetu. Používají ho operační systémy počítačů v síti pro odesílání chybových zpráv - například pro oznámení, že požadovaná služba není dostupná nebo že potřebný počítač nebo router není dosažitelný. Tohoto protokolu také využívá program ping (Packet InterNet Groper), jenž posílá ICMP zprávy typu „Echo Request“ a očekává příjem zprávy typu „Echo Reply“, aby určil, zda je cílový počítač dosažitelný a jak dlouho paketům trvá, než se dostanou k cíli a zpět. Z výsledků lze také získat další parametr sítě, kterým je ztrátovost na dané lince (respektive na dané trase k prvku). Mezi další parametry, které je velice výhodné sledovat patří průtok jednotlivými linkami. Tyto průtoky je nejvýhodnější sledovat na páteřních (případně i lokálních) switchích, pokud podporují protokol SNMP. Dále je vhodné sledovat informace vytížení směrovačů, využití jejich paměti a podobně. Obzvláště pokud jsou síťové prvky přetíženy nebo poddimenzovány je na průběhu vytížení možné tuto závadu snadno odhalit.

Pokud jsou v síti použity bezdrátové prvky je dále vhodné sledovat úroveň signálu, případně také šumu, a to z důvodu snadného odhalení rušení. Některé bez-

drátové prvky opět umožňují tato data získávat pomocí SNMP, u jiných prvků je nutno tato data získávat pomocí skriptů volaných vzdáleně ze strany dohledového systému.

1.4 Ostatní prostředky

Pro realizaci dohledového systému je zapotřebí dalších prostředků jako je databázový server, web-server a v neposlední řadě také vhodný programovací jazyk. Volbou a popisem jazyka se zabývá dále kapitola 2. Web-server je podstatný pro prezentaci získaných dat více uživatelům a to i ze vzdálených lokalit. Měl by nabízet možnosti dynamicky tvořených webových aplikací a k tomuto vhodný jazyk. Navíc je v určitých aplikacích nutné spouštění některých programů umístěných na serveru a samozřejmě také možnost čtení externích souborů ze serveru. Těmto požadavkům vyhovuje většina webových serverů s použitím PHP (rekurzivní akronym pro „PHP: Hypertext Preprocessor“). Více o PHP je uvedeno v kapitole 2.2.

Při vytváření webové aplikace, která načítá nebo ukládá nějaká data, je jedním z nejdůležitějších prvních kroků správný návrh datových struktur, ve kterých budou tato data uchována. Pro ukládání dat jsou nejvhodnější databáze. K webovým aplikacím totiž může přistupovat více uživatelů najednou a např. při ukládání dat do souborů se nevyhneme nutnosti tyto soubory např. správně zamykat. Kromě toho databáze nabízí nesrovnatelně větší komfort při jakékoliv manipulaci s daty. Navíc při vhodném uspořádání databáze je celý výsledek daleko rychlejší a výkonnější.

2 PROGRAMOVACÍ JAZYK

Pro realizaci dohledového systému je nutné zvolit vhodný programovací jazyk. Volba jazyka se samozřejmě odvíjí od nároků na systém kladených. V dnešní době je také důležité vyhnout se také více či méně svazujícím licenčním podmínkám tvůrců daného jazyka. Jazykem, který se v současné době často upřednostňuje je Java. Java však není jediným jazykem, který je nezávislý na operačním systému. Mnoho programovacích jazyků je navrženo tak, aby programy při dodržení určitých omezení bylo možno provozovat na různých operačních systémech. Alternativou pro klasické programovací jazyky, kde je nutno program nejprve kompilovat (přeložit do strojového kódu), jsou takzvané interpretové (respektive skriptovací) jazyky. U toho přístupu proběhne kompilace části programu nebo pouze jeho části pouze v případě využití. Poté je nutné aby byl portován na operační systém pouze interpretr daného jazyka a je možné námi vytvořený program využívat na různých operačních systémech. Jedním z velice dynamicky se rozvíjejících jazyků je také Python, který byl pro tento projet zvolen pro tvorbu aplikační vrstvy.

Pro tvorbu webových aplikací, však Python v současné době není nejvhodnějším a to pouze proto, že není obvykle instalováno rozšíření webového serveru pro stránky psané v Pythonu. S přihlédnutím k této skutečnosti byl jako jazyk pro tvorbu webových stránek jazyk PHP, jenž je v současné době velice rozšířen.

2.1 Python

Python je dynamický interpretovaný jazyk. Jeho možnosti jsou velmi rozsáhlé. Python byl navržen tak, aby umožňoval tvorbu rozsáhlých, plnohodnotných aplikací (včetně grafického uživatelského rozhraní, webových aplikací apod.).

Python je hybridní jazyk (nebo také více-paradigmatický), to znamená, že umožňuje při psaní programů používat nejen objektově orientované paradigma, ale i procedurální a v omezené míře i funkcionální, podle potřeby. Python má díky tomu vynikající vyjadřovací schopnosti. Kód programu je ve srovnání s jinými jazyky krátký a dobře čitelný. K jeho čitelnosti jednoznačně přispívá nutnost dodržování správného odsazení příkazů dle struktury programu.

K význačným vlastnostem jazyka Python patří jeho jednoduchost z hlediska učení. Bývá dokonce považován za jeden z nejvhodnějších programovacích jazyků pro začátečníky. Tato skutečnost je dána tím, že jedním z jeho silných inspiračních zdrojů byl programovací jazyk ABC, který byl jako jazyk pro výuku a pro použití začátečníky přímo vytvořen. Python ale současně bourá zažitou představu, že jazyk vhodný pro výuku není vhodný pro praxi a naopak. Podstatnou měrou k tomu přispívá čistota a jednoduchost syntaxe, na kterou se při vývoji jazyka hodně dbá.

Význačnou vlastností jazyka Python je produktivnost z hlediska rychlosti psaní programů. Týká se to jak nejjednodušších programů, tak aplikací velmi rozsáhlých. U jednoduchých programů se tato vlastnost projevuje především stručností zápisu. U velkých aplikací je produktivnost podpořena rysy, které se používají při programování ve velkém, jako jsou například přirozená podpora prostorů jmen, používání výjimek, standardně dodávané prostředky pro psaní testů (unit testing) a dalšími. S vysokou produktivností souvisí dostupnost a snadná použitelnost široké škály knihovních modulů, umožňujících snadné řešení úloh z řady různých oblastí.

Python se snadno vkládá do jiných aplikací (tzv. embedding), kde pak slouží jako jejich skriptovací jazyk. Tím lze kompilovaným aplikacím psaných v klasických programovacích jazycích dodávat chybějící pružnost. Jiné aplikace nebo aplikační knihovny mohou naopak implementovat rozhraní, které umožní jejich použití v roli pythonovského modulu. Jinými slovy, pythonovský program je může využívat jako modul dostupný přímo z jazyka Python. Tím je tedy docíleno slučování výhod jednotlivých programovacích jazyků, což může výrazně urychlit a zefektivnit výslednou aplikaci.

Jednoznačně programování v Pythonu si klade velký důraz na produktivitu práce programátora. Myšlenky návrhu jazyka jsou shrnuty ve filosofii Pythonu, která je dostupná na oficiálních stránkách projektu [2].

2.2 PHP

PHP je v současnosti velmi rozšířená technologie umožňující snadné programování na straně serveru (server-side programming). Toho lze využít k tvorbě různých interaktivních webových stránek. Stručně je možné říci, že skript napsaný v PHP je proveden na serveru podle zadaných kritérií a výsledek je odeslán volajícímu počítači stejným způsobem, jakým se odesílají běžné HTML (značkovací jazyk pro hypertext – HyperText Markup Language) stránky. Jakmile je však stránka načtena klientem, pomocí PHP ji již není možné dále měnit, což vyplývá z principů funkce webových prohlížečů.

K těmto účelům jsou vytvořeny programovací jazyky jako např. JavaScript. Výhoda těchto programovacích jazyků spočívá v tom, že není třeba pro změnu stránky neustále obnovovat obsah stránky. Jejich veliké omezení však spočívá nejen v možnostech, ale také v jejich (ne)bezpečnosti. Protože skripty se vykonávají přímo na počítači uživatele, mohl by potenciální útočník bez problémů vykonat nebezpečný kód. Z tohoto důvodu jsou například v JavaScriptu vypnuty funkce, které by přímo pracovaly se soubory na harddisku. Díky tomu se JavaScript využívá spíše pro okamžitou úpravu CSS kódu či jako "doplněk" při vytváření webové grafiky. I když

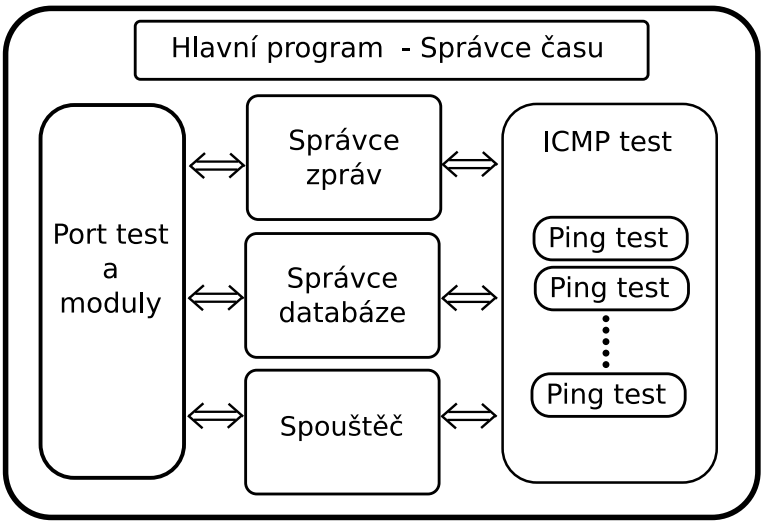
v současné době se velice často využívá vhodných kombinací PHP a JavaScriptu pro tvorbu dynamických stránek. Zde je často využito JavaScriptu k volání konkretního PHP skriptu, který poté vrací ať už klasický HTML kód či dnes často XML.

PHP je programovací jazyk dále umožňující procedurální i objektově orientované programování. Znalost objektově orientovaného programování tedy může být při práci v PHP výhodou, není však nutnou podmínkou. PHP také patří mezi jazyky, kde například není nutné předem definovat typ proměnných, navíc jakákoli proměnná může kdykoli změnit svůj typ.

3 REALIZACE APLIKACE

Vzhledem k nutnosti použití různých a technik přístupu k návrhu a realizaci jednotlivých služeb, které dohledový systém nabízí, je jeho realizace rozdělena na části. První částí je aplikace vykonávající samotné testování. Tato aplikace, které byl dán název NetScan je popsána v kapitole 3.1. Druhou částí je webová aplikace dohledového systému zajišťující přívětivé zobrazení a ovládání dohledového systému. Webová aplikace je složena z klientské sekce a sekce dohledu sítě (podrobněji viz. kapitola 3.2). Testovací a webová aplikace jsou výsledně propojeny přes velmi důležitou třetí část systému, která je tvořena databází (viz. kapitoly 3.4 a 3.3).

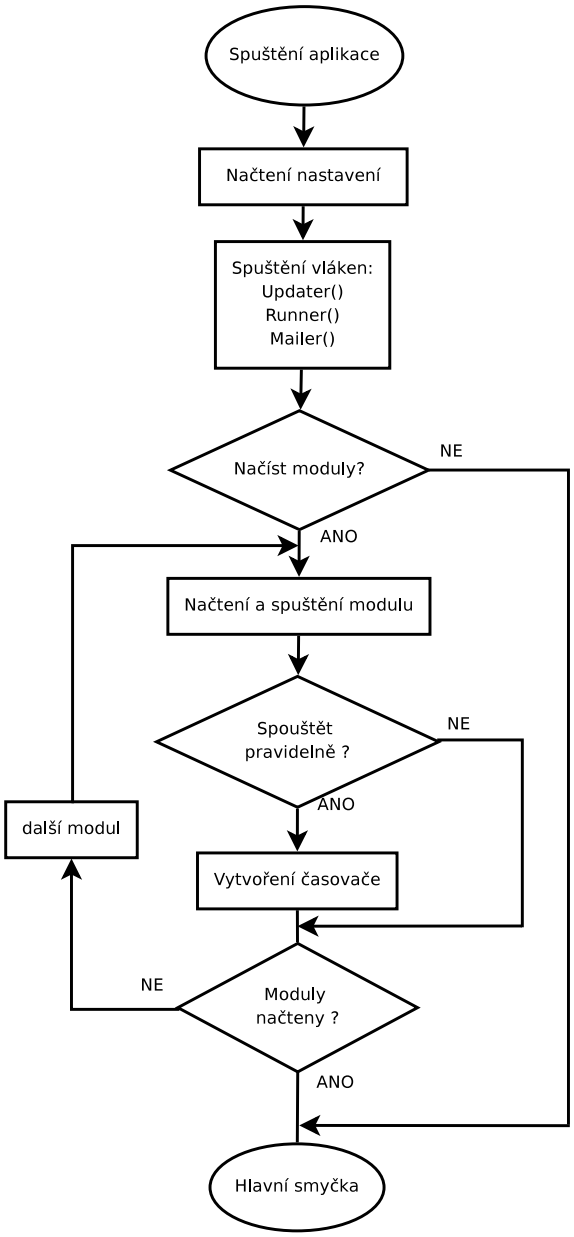
3.1 Testovací aplikace NetScan



Obr. 3.1: Blokové schéma jádra testovací aplikace

Testovací aplikace má za úkol realizovat testovací procedury a získávat data ze vzdálených síťových prvků. Je tedy hlavním zdrojem informací, které jsou předávány databázi. U větších počítačových sítí, které obsahují více síťových prvků je zřejmé, že aplikace musí provádět mnoho testů a obsluhovat mnoho spojení pro získání požadovaných dat. Je tedy nutné, aby testovací aplikace dosahovala vysokého výkonu v požadovaných úkonech. Testovací aplikace je navržena jako co nejjednodušší, neboť k ní není přímo přistupováno jinými procesy ani uživateli (vyjma prvotního nastavení aplikace, jejího spuštění a případného znovu načítání nastavení, či ukončení), což velmi zjednodušuje výsledný kód a tím přispívá ke zvýšení výkonnosti aplikace. Výslednou programovou strukturu je možné vidět na obrázku číslo 3.1. Zde je vidět rozdělení programu na části. Ve skutečnosti každý blok představuje

vlákno obsluhující příslušnou činnost. Na obrázku 3.2 je vidět zjednodušený vývojový diagram spuštění celé aplikace. Nejprve je inicializováno nastavení a v případě OS (operační systém – Operating System) Linux je nastaveno obsloužení signálů (viz. kapitola 3.1.1). Po této inicializaci je spuštěn samotný hlavní program, který již dále zajišťuje běh aplikace.



Obr. 3.2: Diagram spuštění aplikace

Jak je z již řečeného zřejmé další funkcí testovací aplikace je přímé informování prostřednictvím emailových zpráv, aby o případných nesplněních požadavků na testovaná zařízení byl správce informován v co nejkratším čase. Aplikace je navíc navržena pro co největší modulárnost. Tato modulárnost umožní rozšířit množství

funkcí dle libovolných požadavků. Princip modulů je popsán v kapitole 3.1.7. Pokud není využíván žádný přídavný modul, provádí aplikace pouze tzv. ICMP test, který je implementován přímo do struktury aplikace. ICMP test je jen minimálně závislý na testovaných prvcích a je jej možno tedy aplikovat na všechna zařízení reagující na ICMP pakety. Druhým testem realizovaným již uvnitř základní aplikace je testování spojení na TCP porty, pro ověření běhu služeb na daném prvku provozovaných. Další testy jsou realizovány samostatnými moduly, neboť již více závisí na možnostech testovaných prvků a často pro svou náročnost vyžadují zvýšení časového odstupu mezi testy, aby nedocházelo ke zbytečnému vytěžování serveru či sítě.

Jádro aplikace NetScan je tvořeno hlavním programem včetně základních testů a správců zajišťujících většinu základních funkcí aplikace. Na obrázku číslo 3.1 je vidět přehled struktury jádra. Nejprve je spuštěn hlavní program, který v operačním systému Linux přejde do režimu tzv. démona. Tohoto je docíleno vytvořením tzv. fork procesu, ten je tvořen stejným kódem jako jeho rodič. Jedná se podobný princip jako u threadu avšak za podpory operačního systému. Fork procesu (vlastní démon aplikace) je nyní změněn rodič a původní aplikace je uzavřena. Tímto je docíleno provozu aplikace na pozadí.

3.1.1 Hlavní program

Hlavním programem je v tomto případě myšlena třída NetScan. Jedná se o třídu v níž je implementována základní funkce „main“. Tato funkce je mateřskou pro všechny další. V případě OS Linux je navíc tato funkce hlídána při spouštění a Python při jejím prvním spuštění vytvoří tzv. fork, nebo-li nový proces se stejným kódem a původní aplikaci ukončí. Tímto program přechází v „daemona“, což je aplikace běžící na pozadí. Veškeré další informační či chybové výstupy jsou poté zaznamenávány do systémového záznamu (např. syslog).

Dále hlavní program inicializuje nastavení z konfiguračních souborů a vytváří připojení k databázovému systému. Po této inicializaci je spuštěn správce databáze, který zajistí načtení tabulky testovaných prvků

Po inicializaci tabulky prvků hlavní program inicializuje správce zpráv, spouštěč ICMP testů a test portů. Jedná se o samostatná vlákna, která poté pracují nezávisle na hlavním programu (vyjma ukončení) a budou popsány dále. Odkazy na instance těchto tříd jsou nadále součástí hlavního programu aby bylo možné k nim přistupovat. Nakonec hlavní program přejde ve správce času, který zajišťuje pravidelné spouštění testů, případně modulů.

Třída NetScan navíc v případě OS Linux realizuje obsluhu tzv. signálů operačního systému. Jedná se především o zajištění korektního ukončení spojení s databází,

neboť nekorektním ukončením transakcí by mohlo dojít ke ztrátě získaných dat. Obsluhovány jsou především dva signály a to SIGTERM a SIGHUP. Signál SIGTERM je žádost o ukončení aplikace z terminálu. Signál je obsloužen funkcí, která tedy program korektně ukončí a vše zaznamená do systémového záznamu. Signál SIGHUP je v podstatě žádost o nucené znovunačtení konfiguračních souborů, proto je tento signál opět obsloužen. Signály je nutné obsluhovat, neboť jinak by se například oba zmiňované signály chovaly stejně a to vyvoláním okamžitého ukončení programu, což neodpovídá jejich funkcím. Ostatní signály systému jsou buď ignorovány nebo ponechány bez obsloužení nebo obslouženy stejně jako SIGTERM. Podrobněji je obsloužení signálů zřetelné v tabulce 3.1. Jediný signál, který není možné obsluhovat, je SIGKILL a ten tedy vyvolá ukončení bez korektního odpojení od databáze.

Tab. 3.1: Přehled obsluhy signálů OS Linux

Název signálu	Číslo signálu	Význam	Obsluha
SIGHUP	1	"Hangup" - při zavěšení na řídícím terminálu nebo ukončení řídícího procesu.	Znovunačtení konfigurace
SIGINT	2	"Interrupt" - přerušení z klávesnice.	Ignorován
SIGQUIT	3	"Quit" - ukončení z klávesnice.	Ignorován
SIGILL	4	"Illegal Instruction" - neplatná instrukce.	Ukončení a zápis do syslogu
SIGABRT	6	"Abort" - ukončení funkcí abort	Ignorován
SIGFPE	8	"Floating point exception" - přetečení v pohyblivé řádové čárce.	Ukončení a zápis do syslogu
SIGKILL	9	"Kill" - nepodmíněné ukončení procesu	Nelze zachytávat
SIGSEGV	11	Odkaz na nepřípustnou adresu v paměti.	Ukončení a zápis do syslogu
SIGPIPE	13	"Broken pipe" - pokus o zápis do roury, kterou nemá žádný proces otevřenou pro čtení.	Ukončení a zápis do syslogu
SIGALRM	14	Signál od časovače, nastaveného funkcí alarm	Ignorován
SIGTERM	15	"Termination" - signál ukončení	Ukončení a zápis do syslogu
SIGUSR1	30,10,16	Signál 1 definovaný uživatelem	Ignorován
SIGUSR2	31,12,17	Signál 2 definovaný uživatelem	Ignorován
SIGCHLD	20,17,18	Zastavení nebo ukončení dětského procesu	Ignorován
SIGCONT	19,18,25	Pokračování po zastavení	Nepodporován
SIGSTOP	17,19,23	Zastavení procesu	Nepodporován
SIGTSTP	18,20,24	Zastavení znakem "Stop" z terminálu	Ignorován
SIGTTIN	21,21,26	čtení z terminálu v procesu běžícím na pozadí	Ignorován
SIGTTOU	22,22,27	zápis na terminál v procesu běžícím na pozadí	Ignorován

3.1.2 Spouštěč

Spouštěč (třída Runner) je část aplikace, která v podstatě zajišťuje především spouštění vláken u testů či modulů. Jinak řečeno jedná se o správce běžících vláken, neboť je žádoucí, aby celkový počet vláken aplikace byl kontrolován a omezen a to především kvůli prevenci potíží se zahlcováním operační paměti. Aby bylo možné co nejsnadněji možné takového správce realizovat musí obsahovat frontu požadavků na spuštění vláken. V základním nastavení bez přídavných modulů je spouštěč využíván pouze ICMP testem ke spouštění jednotlivých instancí ping testů, jak je podrobněji popsáno v kapitole 3.1.5. Fronta pro ICMP test je samostatná narozdíl od fronty pro

moduly a to z důvodu úspory prostředků a zvýšení efektivnosti. Fronta je v tomto případě reprezentována pouze identifikačními čísly jednotlivých testovaných síťových prvků, jak jsou vytvořeny v databázi (viz. kapitola 3.4).

Jakmile počet aktuálně běžících vláken klesne pod nastavenou maximální hodnotu je odebrán prvně příchozí prvek z fronty a pro něj spuštěno testovací vlákno. Vlákna samozřejmě oznamují své ukončení, aby nebyla nutná náročná kontrola jejich běhu. Pokud by však i přes všechna opatření některá vlákna zablokovala aplikaci jsou hlavním programem násilně ukončována po proběhnutí jednoho cyklu aplikace.

Třída Runner obsahuje metody, které obsluhují požadavky na spuštění vlákna a oznámení vlákna o jeho ukončení. Pro ICMP test jsou přirozeně metody jiné než pro ostatní moduly, neboť moduly musí poskytovat informace o spouštěném vláknu, jako je jeho název apod. Metoda pro žádost o spuštění vlákna ICMP testu nese název `addToRunPing` a je realizována tímto kódem :

```
def addToRunPing (self,nods):
    if len(nods) > 0:        # kontrola délky parametru
        self.ListToRun.extend(nods)    # přidání prvku do fronty
        self.runListFree.set()      # nastavení příznaku
                return
#end def addToPing
```

Jak je vidět jedná se o velmi jednoduchou metodu, která má dva respektive jeden parametr. První parametr je `self` a vychází z volání funkce pomocí „tečkové notace“, která je použita dále. Python tímto způsobem definuje metody tříd. Ukázaná metoda tedy zkontroluje, zda druhý parametr není prázdný, a přidá jej do fronty. Navíc je nastaven příznak přidání do fronty, aby vlákno spouštěče bylo probuzeno, pokud byla fronta prázdná.

Metoda obsluhující moduly je složitější, neboť v první řadě musí být vytvořena fronta pro daný modul. Když je fronta vytvořena průběh metody je již jednodušší. Především však metoda klade nároky na množství předávaných parametrů. Je nutné předávat ukazatel na třídu, která bude spouštěna jako vlákno a maximální počet současně běžících vláken. Metoda je definována následujícím kódem.

```
def addToRun(self,cls,args,maxt=5):
    if len(args) > 0:
        if self.modulesFifos.has_key(cls) :
            self.modulesFifos[cls].append((maxt,args))
    else :
        modulesFifos[cls]=args
    self.runListFree.set()
    return
```

Z kódu je zřejmé, že výchozí hodnota maximálního počtu současně běžících vláken daného modulu je 5. Poté pokud neexistuje je vytvořena položka slovníku obsahující odkaz na obslužnou třídu jako klíč k němuž přísluší tzv. „tuples“ nebo-li

seznam obsahující maximální hodnotu a frontu argumentů pro spuštění vláken. Pokud je fronta již vytvořena je pouze přidán prvek do příslušné fronty podobně jako v případě ICMP testu.

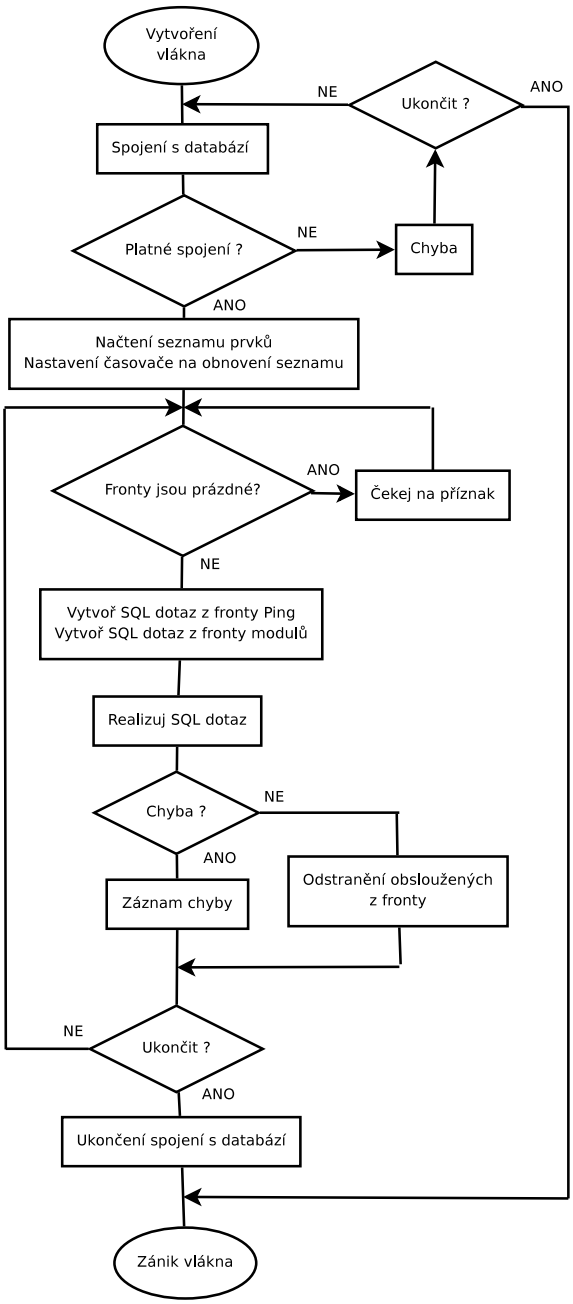
3.1.3 Správce databáze

Správce databáze je velmi důležitým prvkem jádra testovací aplikace, neboť zajišťuje a udržuje komunikaci s databázovým serverem. Je realizováno pouze jedno spojení s databází, která je popsána v kapitole 3.4. Zápis do databáze je sdružován a zápis je prováděn jen jedenkrát v každém cyklu vlákna, tím je dosaženo sdružování zápisů do databáze, které jsou přidány do fronty během jednoho cyklu vlákna. Databáze je takto zatěžována menším množstvím relací a tabulky jsou tak lépe dostupné pro čtení i zápis z webové aplikace. Pokud je fronta dat pro zápis do databáze vyprázdněna je vlákno „uspáno“ a nezatěžuje tak server. Toto pozastavení vlákna je realizováno pomocí čekání na nastavení příznaku. Příznak je nastaven funkcí, která přidává data do fronty zápisu.

Pro ICMP test je opět použita odlišná realizace. Pro potřeby ICMP testu je ve správci databáze implementována speciální fronta zpráv, která je vždy zpracovávána speciální funkcí. Tato funkce analyzuje data a rozhoduje zda daný bod sítě, jemuž data přísluší, splňuje (případně nesplňuje) mezní parametry. Pokud bod nesplňuje požadované parametry je označen za vadný. Na závadu případně na obnovení po závadě je upozorněn správce sítě zasláním výstražné zprávy (odeslání zprávy zajišťuje správce zpráv viz. dále). Výsledná data jsou poté zapsána do databáze. I v tomto případě jsou změny sdružovány pro snížení zátěže databáze.

Čtení z databáze je realizováno metodou, která naopak zajistí okamžité vyřízení dotazu a navrácení výstupních dat ve formě seznamu (tuples). Parametrem metod pro získání dat z databáze a změn v databázi mají parametr, kterým je přímo text SQL dotazu. Tímto řešením je veškerá koordinace dat přenechána na přistupující straně.

Druhou úlohou správce databáze je udržování seznamu síťových prvků s jejich parametry potřebných pro běh aplikace. Již při spuštění správce databáze je načtena tabulka prvků sítě, jež budou testovány pomocí ICMP testu a případně i jinými testy realizovanými moduly. Toto „přednačtení“ je realizováno, aby nebyl nadbytečně zatěžován databázový stroj, neboť jinak by každý jednotlivý test představoval SQL dotaz. Tabulka prvků je převedena na slovník, kterému se někdy též říká asociativní pole. Zde je každý prvek reprezentován svým číselným identifikátorem, pomocí kterého se dále odkazuje na data k němu příslušející. Tato data jsou již ve formě klasického jednorozměrného pole. Takto uspořádaná tabulka již dále umožňuje jednoduchý přístup k potřebným datům. Navíc je tabulka pravidelně po



Obr. 3.3: Vývojový diagram správce databáze

60 vteřinách obnovována z databáze (zajištěno hlavním programem). Přístup k tabulce prvků není realizován metodami, ale přistupuje se k němu přímo. Jedná se sice o porušení zapouzdření, python však toto umožňuje a v daném případě zvyšuje tato realizace výkon celé aplikace. Navíc toto pole je určeno pouze pro čtení takže není toto porušení zapouzdření nikterak závažné.

Na obrázku 3.3 je vidět zjednodušený vývojový diagram vlákna. Vzhledem k odlišné realizaci ICMP testu je opět nutné zpracovávat frontu jeho dat odlišně.

3.1.4 Správce zpráv

Správce zpráv je samostatné vlákno (třída `Mailer`), které zajišťuje rozesílání informačních zpráv pro správce sítě. Zprávy jsou rozesílány jako emailová zpráva zasílaná pomocí SMTP protokolu. Je zde využit modul pythonu smtplib. Tento modul zajišťuje komunikaci se SMTP serverem, který je definován v nastavení testovací aplikace (pokud není zadán je použit localhost).

Správce zpráv samozřejmě umožňuje odesílaní výstražných zpráv více uživatelům a to dle tabulku uživatelů v databázi (viz. kapitola 3.4). V této tabulce je navíc definováno, které dny v týdnu a v jakém časovém období ve dni budou na adresu uživatel zasílány výstražné zprávy. Toto nastavení se projeví i na případné zprávy zasílané od rozšiřujících modulů. Také je možné zvolit zda zprávy generované mimo zasílací čas jsou ukládány a zaslány v nejbližším povoleném termínu nebo jsou ignorovány. V případě shromažďování zpráv je nutné vytvoření fronty pro každého uživatele zvlášť. Vzhledem k faktu, že vlákno je při prázdné vstupní frontě pozastaveno a čeká na nastavení příznaku přidání do fronty, je navíc nutné vlákno „probudit“ v době začátku zasílacího intervalu uživatele. K tomuto účelu je využit objekt `scheduler` (plánovač), jenž má nastaven vypočtený čas pro zaslání zprávy. V tento čas je spuštěna metoda pro odeslání zprávy, resp. všech zpráv ve frontě. Zprávy zasílané systémem by měli být co nejkratší, aby bylo možné například jejich přesměrování na mobilní telefon v podobě SMS (krátká textová zpráva šířená pomocí telefonu – Short Message Service) zpráv.

Pro potřeby ICMP testu je i v tomto bloku implementována speciální fronta zpráv, do které jsou přidávány pouze hodnoty nikoli celý text zprávy. V současné době jsou pro ICMP test podporovány dva typy zpráv - zpráva o poruše a zpráva o obnovení z poruchy. Zpráva o poruše obsahuje vždy název prvku, IP adresu, odezvu a ztrátovost. Zpráva o obnovení obsahuje opět název, odezvu, ztrátovost a délku výpadku, narozdíl od zprávy o poruše neobsahuje IP adresu, aby došlo k úspoře textu. Zprávy stejného typu jsou v jednom testovacím cyklu sdružovány, aby nedocházelo k odesílání velkého množství zpráv například v případě výpadku páteřního síťového prvku, což znemožní komunikaci s částí celé sítě. Takto sdružené zprávy jsou poté odeslány pomocí protokolu SMTP na mail server. Pro ICMP test je navíc zavedena speciální varianta upozorňování uživatele na začátku upozorňovacího období. Jedná se variantu, kdy je uživateli zaslán seznam aktuálně vadných prvků. Tato varianta je ideální například v případě střídání správců sítě ve směnách. Správce je na začátku své směny upozorněn na aktuální stav sítě.

Přídavné moduly ke správci zpráv přistupují pomocí metody `sendMessage` s následující deklarací.

```
def sendMessage(self,text,user='all',odloz=0):
```

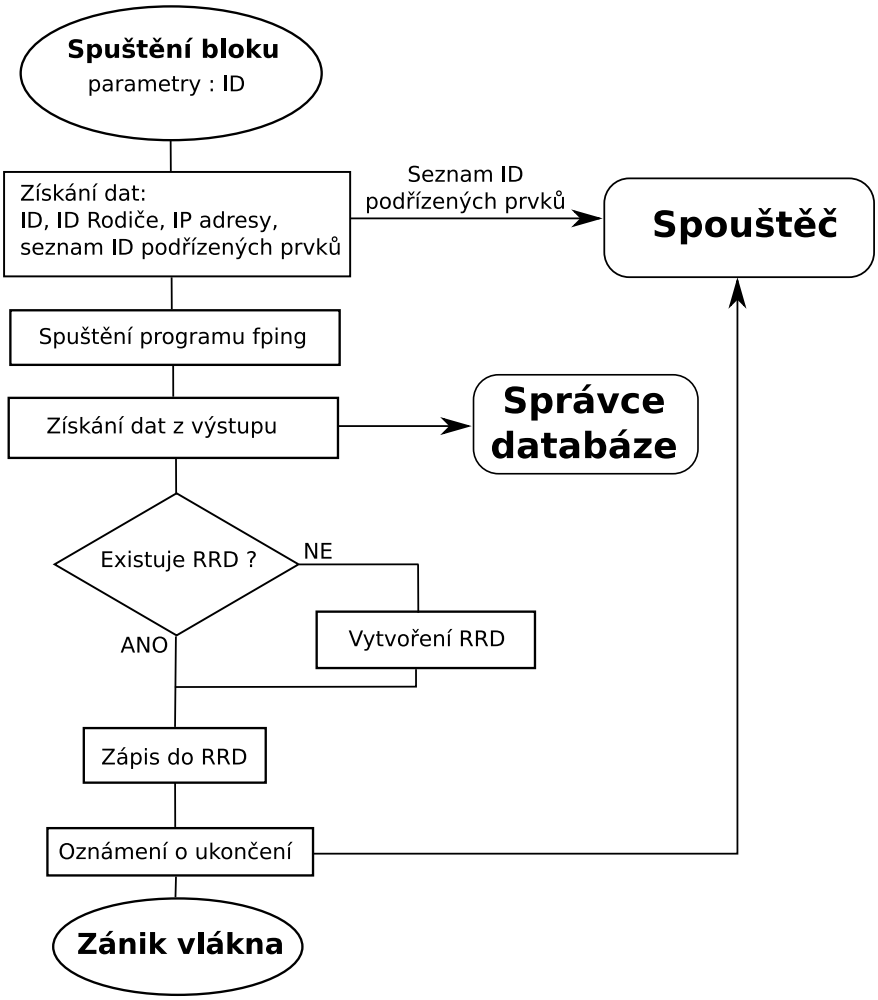
Neboť se jedná o metodu třídy, je prvním parametrem ukazatel na třídu (vychází z „tečkové notace“). Druhý parametr je zasílaný text. Třetím parametrem je název uživatele, kterému je zpráva určena. Pokud není uživatel zadán je zpráva zaslána všem uživatelům. Posledním parametrem je nastavení odkládání zprávy do fronty, pokud uživatel aktuálně zprávy nepřijímá. Ve výchozím stavu zprávy odkládány nejsou a jsou tedy doručeny pouze uživatelům, kteří zprávy aktuálně přijímají.

3.1.5 ICMP test

ICMP test je základním testem dohledového systém a je proto implementován do jádra. Jeho hlavním úkolem je spouštění bloků ping, jak je vidět na obr. 3.1, s příslušnými parametry a výsledky testu odeslat k dalšímu zpracování.

Blok ping zajišťuje spuštění programu ping (případně jeho alternativy) a dohled nad během programu. Po dokončení programu blok ping zpracovává výstup programu, kde získá počet odeslaných paketů, počet přijatých paketů včetně minimální, maximální a průměrné časové odezvy na žádost o „echo reply“. Pro potřeby přehlednosti je také zaznamenáván stav nadřazeného prvku sítě. Parametry programu ping je možné nastavit v konfiguraci celého systému, je však doporučeno ponechat přednastavené. Přednastavené parametry definují počet odesílaných ICMP paketů typu echo request na 50, čas mezi odesíláním paketů na 1 ms a maximální čas odezvy na 1000 ms. Takto zvolené parametry jsou optimálním kompromisem mezi přesností výpočtu a nadbytečným zatížením sítě. K vlastnímu testovaní je v systému možné použít jednu ze tří variant. První variantou je vlastní zjednodušená implementace napsaná přímo v pythonu. Jedná se o výchozí variantu, její výhodou je především nezávislost na OS. Jsou pouze požadována práva superuživatele pro surový (RAW) přístup k soketu. Druhou variantou je spouštění programu „ping“. Optimalizace je provedena na program ping z OS Linux, neboť například program ping v OS Windows neposkytuje dostatečně širokou škálu nastavení. Využití programu ping vykazuje vyšší výkon než realizace v pythonu. Poslední variantou je program fping, který je původně určen pro testování mnoha bodů postupně za sebou. V dohledovém systému je však využíván k testování vždy právě jednoho prvku a to z důvodu větší časové efektivity, neboť bylo zjištěno, že spouštění jednotlivých testů i za sebou vykazuje kratší délku zpracování než využití hromadného zadání programu. Výhodou programu fping je jeho portace do více OS.

Dalším úkolem bloku ping je zapisovaní získaných dat do Round-Robin databáze popsané v kapitole 3.3. Vzhledem ke skutečnosti, že výsledkem ping testu je 6 hodnot pro každý prvek sítě, jedná se v souhrnu o větší množství dat. Zápis proto není realizován pomocí správce databáze, aby nedocházelo ke zbytečnému předávání velkého množství dat mezi jednotlivými bloky testovací aplikace. Navíc zápis



Obr. 3.4: Blokové schéma funkce bloku ping

do Round-Robin databáze je velmi rychlý a nezpomaluje tedy celou operaci testování, čímž je jeho začlenění do bloku ping optimálním kompromisem mezi rychlostí a náročností na zdroje operačního systému jako jsou paměť či výkon procesoru. K drobnému zpomalení celého systému může docházet pouze v případě vytváření nových souborů Round-Robin databáze. Tato situace je však dorovnána v dalších testovacích cyklech , takže není nutné provádět jakoukoli optimalizaci při vytváření databází. Navíc okamžitý zápis do Round-Robin databáze zachovává časovou přesnost dat, neboť opožděný zápis by posouval časovou osu databáze.

Posledním úkonem, který blok ping vykonává je získání seznamu podřízených prvků a jejich následné zařazení do fronty ke zpracování. Přehled celé funkce je nejlépe zřetelný z blokového schématu funkce na obrázku č. 3.4.

Aby bylo možné provádět testy pro mnoho bodů v co nejkratším čase je nutné blok ping spouštět vícekrát. Množství instancí bloku ping je opět možné nastavit v konfiguraci systému, výchozí hodnota je 20 samostatné běžících instancí bloku

ping. Toto nastavení umožňuje při výchozím nastavení parametrů programu ping otestovat přibližně 1000 bodů a to i v případě nedostupnosti všech bodů, kdy je nutné čekat na maximální odezvu bodu (přibližně 1200 ms). Množství aktuálně spuštěných instancí bloku ping kontroluje a spravuje blok spouštěč (viz. obr.3.1) popsaný v kapitole 3.1.2. Spouštěč navíc obsahuje frontu identifikátorů bodů, které má systém dále testovat.

3.1.6 Port test

Port test je jednoduchý test otevřených TCP portů. Pokud na daném portu naslouchá aplikace (např webový server) a port není filtrovaný je možné se k němu připojit. Tohoto jevu je možné využít k jednoduché kontrole běžících služeb. Výsledky testu jsou zaznamenávány pouze ve formě výpadků s jejich délkou do SQL databáze. K samotnému testování je na výběr z pythonové realizace testování portů, či využití externího programu nmap. Využití programu nmap nabízí navíc možnost testování UDP portů.

Test portů realizovaný v pythonu je založen na nemožnosti otevření socketu na neotevřený či filtrovaný port. Pro testování je nastaven časový limit 10 sekund na spojení, po uplynutí časového limitu je port považován za nedostupný.

Často jeden síťový prvek nabízí více služeb na příslušných portech. Z toho vyplývá, že testovaných portů může být mnohem více než prvků v síti. Z tohoto důvodu je nutné test opět spouštět paralelně, podobně jak tomu je u ICMP testu. Vzhledem k delšímu časovému limitu je navíc nutné navýšení počtu vláken. Ve výchozím nastavení je maximální počet vláken pro testování portů třicet.

3.1.7 Struktura modulu

Pro účely rozšiřování dohledového systému o další funkce je systém navržen pro přidání tzv. modulů. Modul je samostatný kód, který je spouštěn jádrem dohledového systému. Jádro testovací aplikace nabízí pro moduly několik základních služeb. První službou, kterou poskytuje konkrétně správce databáze, je aktualizovaný seznam prvků sítě. Samozřejmě také správce databáze poskytuje metody pro práci obecně s databází (zadávání SQL dotazů). Další nabízenou službou je možnost odesílání zpráv uživatelům dohledového systému. Poslední je služba spouštěče vláken, který nabízí možnost koordinace spouštění vláken modulu s omezením maximálního počtu vláken. Přídavný modul si sám řídí časy, ve kterých je spouštěn, neboť může být pro některé aplikace nutné spouštět testování méně často, než je základní perioda ICMP testu. Je také možné zajišťovat spouštění modulu v násobcích základního cyklu hlavního programu.

Modul testovací aplikace musí splňovat několik požadavků, aby byl akceptován a spuštěn hlavním programem. Modul musí být realizován v podobě třídy, která má vlastnosti vlákna, aby jeho kód běžel nezávisle na hlavním programu. Tím je docíleno nenarušování běhu hlavního programu. Z funkcí vlákna je hlavním programem využívána především metoda `start()` a `joint()` určené ke spuštění a ukončení vlákna. Přídavný modul musí také obsahovat metodu `__init__` (obdoba konstruktoru např. v Javě) jejímž parametrem s názvem `netscan` musí být odkaz na třídu NetScan. Instance třídy přídavného modulu definovaného třídou s názvem `SNMP` je z hlavního programu totiž volána takto :

```
import SNMP # načtení kódu modulu
modul = SNMP(this,args) #vytvoření instance modulu
modul.start() #spuštění vlákna modulu
```

V tomto případě je parametrem `this` instance třídy NetScan a parametrem `args` jsou parametry modulu zadané v konfiguraci testovací aplikace. Ve skutečnosti není možné výše uvedený kód přesně takto použít, neboť hlavní program musí být nezávislý na názvu přídavného modulu, je proto nutné nahradit klasický `import`. Za tímto účelem je použit modul `imp` pythonu, který umožňuje importovat modul s názvem dle proměnné. Nejprve je modul hledán a pokud je nalezen je zaveden a přidán do seznamu zavedených modulů.

Aby bylo možné data získávaná pomocí testovacího modulu prezentovat, je nutné realizovat také modul webové aplikace (viz. kapitola 3.2.1).

3.1.8 Modul SNMP

Mnoho síťových prvků dnes podporuje možnost získávání informací pomocí protokolu SNMP (jednoduchý protokol pro správu – Simple Network Management Protocol). Pomocí tohoto protokolu je možné získávat mnoho užitečných informací především z klíčových prvků sítě, jako je množství přenesených paketů či bajtů, hodnota vytížení procesoru atd. Jednotlivé prvky se liší poskytovanými informacemi. Jak již bylo popsáno v kapitole 1.2.1 je využíváno MIB databází protokolu. Tyto databáze obsahují jak název tak číselné označení jednotlivých položek. Pro získávání informací protokolem SNMP byl pro python vyvinut modul `pysnmp` [6], který je vlastně jakýmsi frameworkem pro práci s tímto protokolem. Zmiňovaného modulu využívá také modul pro testovací aplikaci. Aplikační modul je realizován opět jako vlákno získávající informace ze síťových prvků uvedených v databázi ve speciální tabulce modulu (podrobněji viz. kapitola 3.4). Vzhledem k náročnější realizaci získávání dat pomocí protokolu SNMP má vlákno modulu ve výchozím nastavení prodloužený cyklus a to na interval 3 minut. V modulu jsou rozlišovány tři typy získávaných informací.

Prvním typem jsou informace z jakýchkoli čítačů (např. čítače datových průtoků). Tento typ informací je vyobrazován zpravidla graficky a je tedy proto zaznamenáván do Round-Robin databáze, jejíž název a potřebné parametry jsou opět získány z databáze, do níž je zadává uživatel skrze webovou část modulu. Data jsou do RRD zaznamenávána v typu `COUNTER` hned po jejich získání protokolem SNMP, aby byla časová osa co nejpřesnější. Webová část pak ze získaných dat vytváří grafy.

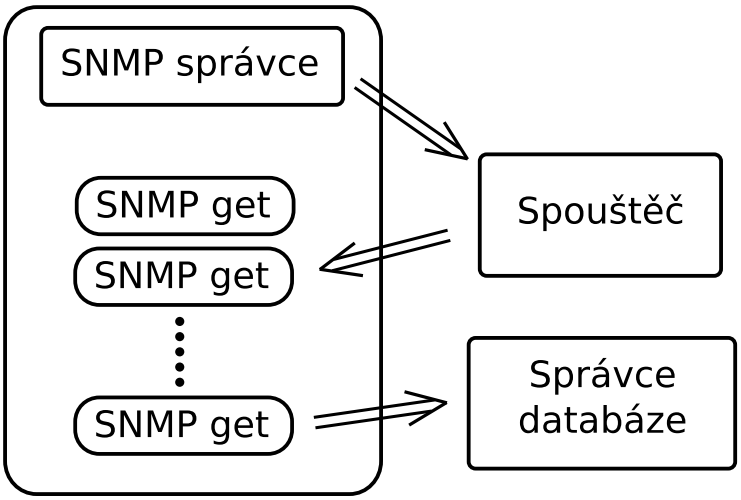
Druhým typem informací jsou aktuální hodnoty veličin jako například využití času procesoru. Tato data jsou také z pravidla prezentována graficky a tak je opět využito Round-Robin databáze. Oproti čítačovým datům je v tomto případě nutné použít jiného datového typu a zvolit agregační funkci. Tyto parametry jsou opět získávány z SQL databáze.

Třetím typem dat jsou textové či jiné informace. Tato data je již z principu nutné zaznamenávat pomocí SQL databáze. Například je možné takto získávat MAC (jedinečný identifikátor síťového zařízení – Media Access Control) adresy příslušící IP adresám z ARP záznamů, pokud směrovač na hranici segmentu sítě podporuje možnost získávat tato data protokolem SNMP. U tohoto typu dat je obecně složité definovat universální rozhraní pro libovolné hodnoty. V současné době je proto implementováno pouze rozraní pro získávání zmiňovaných MAC adres. Takto získané adresy jsou přiřazeny konkrétní IP adrese v databázi se zaznamenáním posledního času, kdy byla adresa získána. Tato data jsou velmi užitečná pro kontrolu zneužívání IP adres a neoprávněných připojování.

Pro vlastní realizaci získávání dat je opět vytvořeno vlákno, které komunikuje vždy právě s jedním síťovým prvkem. Parametrem vlákna je seznam OID (identifikátor objektu – Object Identifier) z příslušné MIB včetně označení typu dat. Tento typ realizace umožňuje opět běh více vláken neboť komunikace SNMP protokolem může být časově náročná v závislosti na odezvě síťového prvku. Navíc je takto realizován pouze jeden dotaz do seznamu prvků. Vlákna jsou spravována spouštěčem z jádra testovací aplikace. Hlavní vlákno modulu zajišťuje tedy především obnovu seznamu zjišťovaných OID a doplňování fronty spouštěných vláken. Ostatní úkony jsou realizovány již jednotlivými vlákny.

V modulu je pro čítače přenesených dat a počtu paketů vytvořena speciální funkce, neboť tato data jsou nejčastější a také dostupná na většině zařízeních. Společně s těmito daty je také zaznamenáváno zda konkrétní rozhraní prvku je aktivní, neboť tento fakt je při získávání také nutno ověřit.

Blokový diagram modulu je vidět na obrázku 3.5. Části řešící jednotlivé typy získávaných informací jsou v obrázku označeny. Rozhodnutí o použití konkrétní funkce pro obsluhu je opět získáno z parametrů uložených v SQL databázi.



Obr. 3.5: Blokové schéma modulu SNMP

3.1.9 Modul TCPd

Modul TCPd je nazván podle progamu tcpd, který je využíván k obsluze příchozích spojení na vybrané TCP porty v OS Linux. Program je zpravidla volán síťovým „superdaemonem“ a jeho parametrem je příkaz jehož výstup je zaslán po síti zpět iniciátorovi spojení. Tímto způsobem je možné realizovat obecné získávání a zaznamenávání dat do grafů případně SQL databáze. Realizace modulu je velmi jednoduchá. v podstatě se jedná pouze o vytvoření spojení na příslušný port prvku sítě a zaznamenání dat do grafu či databáze. Pro modul je opět vytvořena tabulka SQL databáze v níž jsou uloženy parametry testu a parametry pro způsob uložení.

3.2 Webové aplikace systému

Webová aplikace je vzhledem charakteru celého systému rozdělena do dvou částí. První částí je webové rozhraní pro dohled nad sítí a druhou částí je klientská databáze. Obě tyto části jsou propojeny, aby bylo možné rychle zjišťovat návaznost jednotlivých dat. Provázanost je realizována pomocí odkazů na konkrétní adresu dané části nebo jsou načtena příslušnádata jako nápovědný text. Pro obě části jsou společné principy zabezpečení, které jsou popsány v kapitole 3.2.3.

Pro tvorbu webové aplikace byla vybrána kombinace jazyka PHP na straně serveru [7] a JavaScriptu na straně webového prohlížeče [11]. Je kladen důraz na zmenšení počtu přenačítání celého obsahu stránky. Pro tuto minimalizaci je využito asynchronní komunikace pomocí javascriptu (často bývá tato komunikace označována jako AJAX). Tento typ komunikace umožňuje načítat pouze požadované objekty a nikoli celou stránku. Tím se snižuje zatížení serveru a navíc se tím celá webová

aplikace přibližuje svým chováním běžným aplikacím používaným na osobních počítačích. Také je umožněno aby bylo možné poskytovat uživateli různé nápovědy a tím zrychlit a usnadnit jeho práci.

Pro přímé testy je použito pythonu a to v podobě mod_python modulu pro webový server Apache [12]. Veškeré testy by bylo také možné realizovat pomocí PHP či CGI a však mod_python byl použit pro větší možnosti zabezpečení proti zneužití.

3.2.1 Dohledová aplikace

Dohledová aplikace slouží především jako rozhraní pro správce sítě. V podstatě převádí data získaná během měření na informace snadno srozumitelné a přehledné pro uživatele. Dohledová aplikace je po vzhledové stránce tvořena dvěma rámy. V první rámu je zobrazován dynamický strom prvků sítě (viz. kapitola 3.2.1). Tento strom odkazuje na statistiky jednotlivých prvků či různé přehledy. Tyto statistiky jsou poté zobrazovány v druhém rámu a obsahují navíc nabídku s nastavením testů pro daný bod. Výsledné zobrazení po načtení aplikace je vidět v příloze č. B.1.

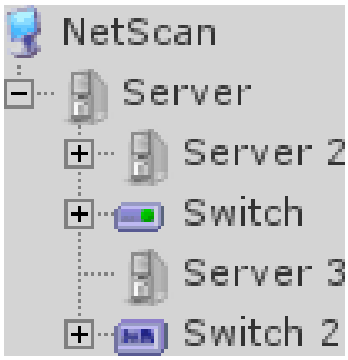
Dynamický strom

Pro přehledné zobrazení struktury sítě slouží tzv. dynamický strom. Strom je vytvořen při načtení webové stránky dohledové části systému. Ve stromu je zobrazena ikona podle typu síťového prvku tak jak byla zvolena při vytvoření prvku či následně změněna v nastavení prvku. S ikonou je zobrazen také stručné pojmenování prvku. Strom by měl odpovídat zjednodušené topologii sítě (v případě více cest není možné realizovat zobrazení stromem). Prvky za nimiž jsou připojené další prvky je možné rozbalit a podprvky tak zobrazit.

Strom je prakticky celý realizován a vykreslen na straně webového klienta pomocí javascriptu. Jedná se o třídu `Tree`, které je serverem předáno pouze pole obsahující srovnaný seznam prvků. Z tohoto seznamu je již javascriptem vytvořen celý strom jako struktura navzájem provázaných objektů obsahující také příslušný HTML kód, jenž je zobrazen prohlížečem. Seznam všech objektů je zaznamenán jako pole v instanci vlastní třídy `Tree`. Třída navíc obsahuje obslužné metody pro funkce stromu (např. zobrazení statistik či rozbalení prvku).

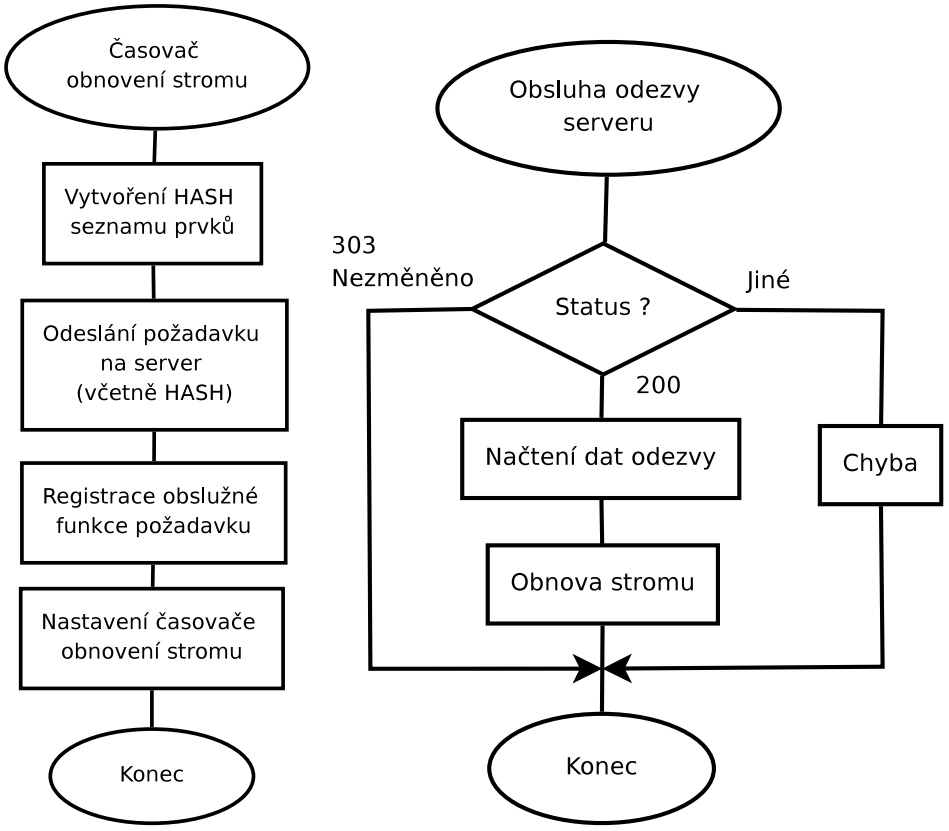
Kořenem stromu je pomyslně aplikace samotná, což je také viditelné na obrázku 3.6, který zobrazuje příklad stromu. Kliknutím na kořen se zobrazí výchozí statistika aktuálně vadných prvků včetně délky výpadku. Interakcí s jednotlivými body je poté základní statistika daného prvku.

Strom je v pravidelných intervalech obnovován. V každém obnovení je nutné načíst seznam prvků vykazujících závadu. Tento seznam je pouze pole identifikačních



Obr. 3.6: Ukázka dynamického stromu

čísel vadných prvků. Těmto prvků je na popředí přídána před standardní ikonu ikona označující vadu a cesta ve stromu k prvku je rozbalena tak, aby byl prvek viditelný pro uživatele. Obnovování vadných prvků je prováděno v intervalu dle nastaveného základního cyklu testovací aplikace.



(a) Dotaz na webový server (b) Obsluha odezvy serveru

Obr. 3.7: Vývojový diagram obnovy dynamického stromu

Dále je nutné, aby strom reagoval na změny, které v něm byly provedeny, respektive na změny provedené v databázi, ze kterých je strom vytvářen. Zde je situace

z hlediska realizace poněkud složitější. Pokud je proveden zásah do databáze, který ovlivňuje strukturu je nutné strom překreslit. Toto překreslení je realizováno opět javascriptem a asynchronní komunikací a je prováděno každé 3 minuty. Identifikace změny je realizována vytvořením tzv. HASHe pomocí hashovací funkce MD5, která je aplikována na původně získaný seznam prvků. Tento hash je zaslán jako součást dotazu serveru, který stejnou hashovací funkcí provede stejnou akci se seznamem prvků, vytvořeného z aktuálních dat v databázi. Pokud se výsledky hashovacích funkcí nerovnají, je nutné strom obnovit a je tedy serverem zpět zaslán nový seznam prvků. V opačném případě je pouze odeslána zpráva o tom, že požadovaná stránka nebyla změněna. Obsluha asynchronního dotazu na straně klienta obnoví data ve stromu pokud obdrží nový seznam či změní čas generování stromu, aby bylo možné z pohledu uživatele kontrolovat, zda je strom aktuální. Pokud by v průběhu bylo spojení se serverem přerušeno, je uživatel informován o chybě spojení se serverem. Celý popisovaný průběh obnovování stromu je zřetelný z obr. 3.2.1, který zobrazuje vývojový diagram. V části (a) obrázku je zřetelné reakce na časovač obnovení stromu. Je zde zaregistrována obslužná funkce asynchronního dotazu na server. Obslužná funkce je poté vidět v části (b) obrázku 3.2.1. Obslužná funkce zjistí stav odezvy z HTTP hlavičky. V případě stavu 303 (stránka nezměněna) neprovádí funkce žádné operace, neboť strom je aktuální. Pokud má stav hodnotu 200 (žádost vyřízena), jsou načtena data z odezvy serveru, strom obnoven a překreslen. Po překreslení stromu je nutné aktualizovat také označení vadných prvků. Pokud má stav jinou hodnotu, je komunikace považována za chybnou a strom je označen za neaktuální.

Statistiky, grafy, přehledy

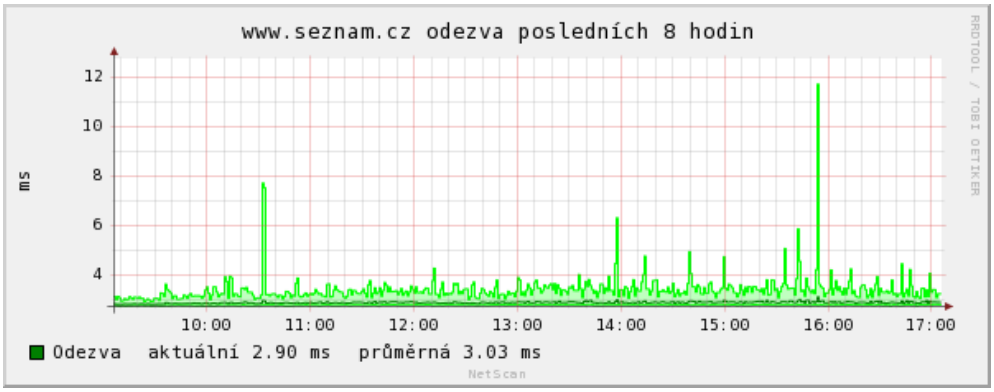
Jak již bylo řečeno výše, webová stránka dohledu obsahuje rám určený pro statistiky a přehledy. Statistiky se skládají z textové části, v níž jsou shrnuty dlouhodobé parametry, zobrazen seznam posledních závad a samozřejmě popis síťového prvku. Textová část obsahuje také nabídku, která umožnuje přepínání statistik, přepínání zobrazení grafů, přechod do nastavení prvku a také možnost jeho přesunutí v rámci hierarchie sítě. Příklad zobrazení statistky včetně grafů za posledních 8 hodin je vyobrazen v příloze B.2.

Grafy jsou tvořeny z dat uložených v RRD, a proto, jak bylo již řečeno výše, je možné využít RRDTool pro snadné vytváření grafů. Za tímto účelem jsou vytvořeny PHP skripty pro každý typ RRD. Prvním typem je RRD obsahující údaje o odezvách příslušného síťového prvku. Zde je generován graf jehož příklad je viditelný na obrázku 3.8. Průměrná odezva je zde zaznamenána tmavě zelenou barvou a rozptyl mezi maximální a minimální odezvou je vyplněn světle zelenou. Skript generující graf

a všechny RRD v celé aplikaci jsou navrženy tak, aby bylo možné vytvářet grafy pro posledních 8 hodin, 24 hodin, 2 dní, 14 dní 1 měsíce a 1 roku. Na tyto délky intervalů jsou uzpůsobeny RRA. Požadovaný interval je definován parametry, kterými je skript volán. Dalšími parametry skriptu jsou ID prvku a volba logarytmického zobrazení. Logarytmické zobrazení umožňuje zmírnění vlivu náhodných maximálních hodnot u odezvy. Ve webovém rozhraní je možné přepínat logarytmické zobrazení kliknutím na graf odezvy.

Dalším typem grafu je zobrazení průběhu ztrátovosti. Zde není nutné vytváření logarytmického zobrazení, a však velice výhodná je zde změna barvy pozadí průběhu v případě závady nadřazeného prvku. Takto vytvořený graf usnadní uživateli hledání závady, pokud je síť kontrolována s odstupem času. V grafu je navíc vyobrazena nastavená maximální přípustná hodnota ztrátovosti

Statistika jednotlivých prvků obsahuje také odkaz na testované služby (porty) síťového prvku. tento odkaz zobrazí jednoduché časové osy pro příslušné časové období s vyznačením výpadků služby (portu). Tento způsob zobrazení byl přehlednost, neboť seznam výpadků je méně vypovídající a navíc, v zobrazení na časových osách je možné vidět propojenost závad jednotlivých služeb (portů).



Obr. 3.8: Příklad grafu průběhu odezvy

Dohledový systém nabízí také přehledy, které je možné řadit dle libovolných kritérií. Základními přehledy jsou přehledy tendence vývoje funkčnosti a přehledy průměrných odezev prvků. Přehled vývoje tendence funkčnosti nabízí uživateli předběžnou informaci zda a jaké síťové prvky nesplňují požadovaná kritéria. Z přehledu je možné odhalit prvky, které obnoví svou funkčnost dříve než je systém vyhodnotí jako vadné. Přehled navíc podává ucelený souhrn aktuálně nefunkčních prvků včetně délky aktuální poruchy. Tento přehled je výchozím zobrazením po načtení webové stránky dohledového systému. Přehled průměrných odezev prvků podává ucelenou a přehlednou informaci o dlouhodobém vývoji průměrných odezev na jednotlivých prvcích sítě. Pomocí toho přehledu je možné sledovat zhoršování odezvy na jednot-

livých spojích a tím předcházet např. přetěžování spojů jejich včasnou náhradou výkonnějšími. Tento přehled navíc obsahuje poslední naměřenou hodnotu odezvy.

Grafy statistiky opět nejsou obnovovány jako celek, ale pomocí asynchronní komunikace jsou obnoveny pouze proměnné části zobrazovaného objektu.

Přímé testy

Dohledová webová aplikace navíc podporuje několik základních přímých testů. Přístup k testům je pomocí kliknutí na IP adresu prvku ve statistice. Tím se zobrazí nabídka všech dostupných přímých testů. Testy jsou prováděny s IP adresou prvku. Na IP adresu je však odkazováno pomocí identifikačního čísla prvku tak, aby se předešlo útokům, které by do parametrů obslužného skriptu vkládaly závadný kód.

První testem je klasické testování odezvy (obdoba příkazu ping). Vzhled i výstup testu je obdobný s programem ping. Pro test je možné nastavit počet paketů, časové rozmezí mezi jednotlivými pakety a délku paketu. Pro časové rozmezí mezi pakety platí omezení na minimální čas 20 ms, který vychází z omezení uživatelských práv a opětovně předcházení útokům. Po vykonání testu je zobrazena statistika testu, která navíc oproti klasickému programu ping vyobrazuje také jitter, který vypovídá o kolísání odezvy.

Dalším testem je takzvané procházení cesty paketu. Jedná se o obdobu programu traceroute, který poskytuje uživateli informace o odezvách směrovačů, přes které paket prochází. Tento test pomáhá kontrolovat aktuální topologii sítě.

Pro směrovače je také připraven test, který vyobrazí maximální zjistitelný počet uživatelů (síťových prvků) v daném segmentu sítě. Test je realizován vysláním zkušebních ICMP paketů na všechny možné IP adresy v dané podsíti, čímž se naplní ARP cache směrovače záznamy všech dostupných prvků. Obsah ARP cache je poté protokolem SNMP získán ze směrovače a zobrazen jako výstup uživateli. Z principu testu vyplývá, že je nutné, aby směrovač podporoval protokol SNMP a umožňoval získávání obsahu ARP cache pomocí tohoto protokolu. U tohoto testu je navíc realizováno propojení s klientskou databází. Pro získané IP adresy jsou načtena data o vlastníkovi, pokud je v klientské databázi uveden.

Všechny přímé testy jsou realizovány pomocí zmiňovaného modulu mod_python webového serveru Apache.

Přídavné moduly

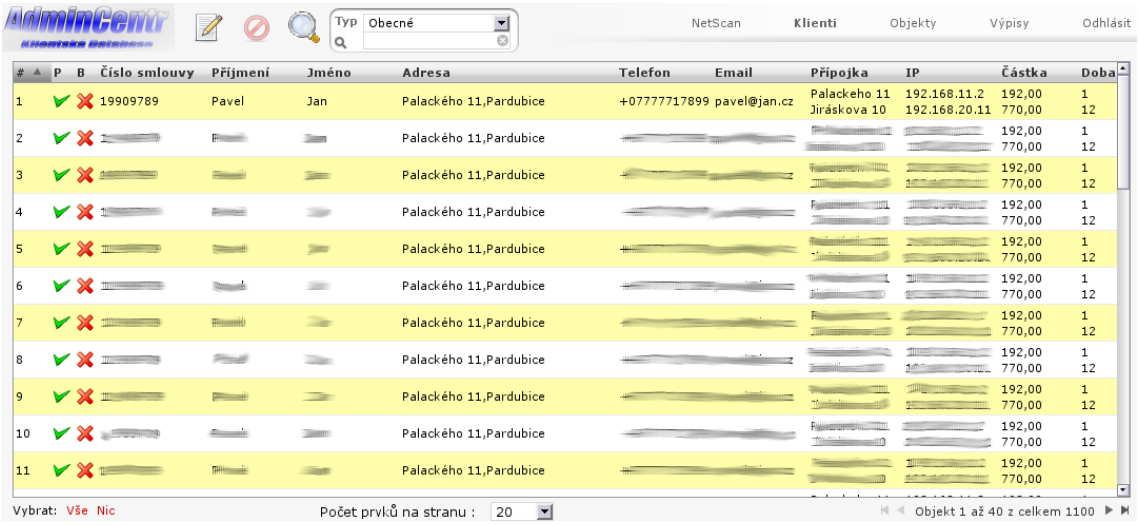
Do webové aplikace, podobně jako v případě testovací aplikace, je možné přidávat funkce a to v podobě přídavných modulů. Moduly jsou navíc nutné pro zobrazování získaných dat pomocí přídavných modulů testovací aplikace a samotného nastavování modulů.

3.2.2 Klientská databáze

Klientská databáze umožňuje přehlednou a efektivní práci s údaji o klientech a službách, které využívají. Klientská databáze nabízí tři základní zobrazení.

Kartotéka

Prvním zobrazením je tzv. kartotéka. jedná se o seznam stručného popisu klientů jak je vidět na obrázku 3.9. Toto zobrazení umožňuje řazení klientů a jejich filtraci dle zvolených parametrů. Je možné rychle vyhledávat dle jedné podmínky zadané do rychlého vyhledávaní v horní části stránky, nebo pomocí ikony rozšířeného vyhledávání zadávat více podmínek. Stránka umožňuje také odstraňovat označené klienty či nové klienty přidávat nebo měnit počet položek zobrazených na jedné stránce. U každého klienta je pomocí asynchronní odezvy navíc získáván seznam podrobností, který je zobrazen jako nápovědný text po najetí myši na konkrétní položku seznamu. Vybráním položky je poté možné přejít do zobrazení a editace podrobností klienta.



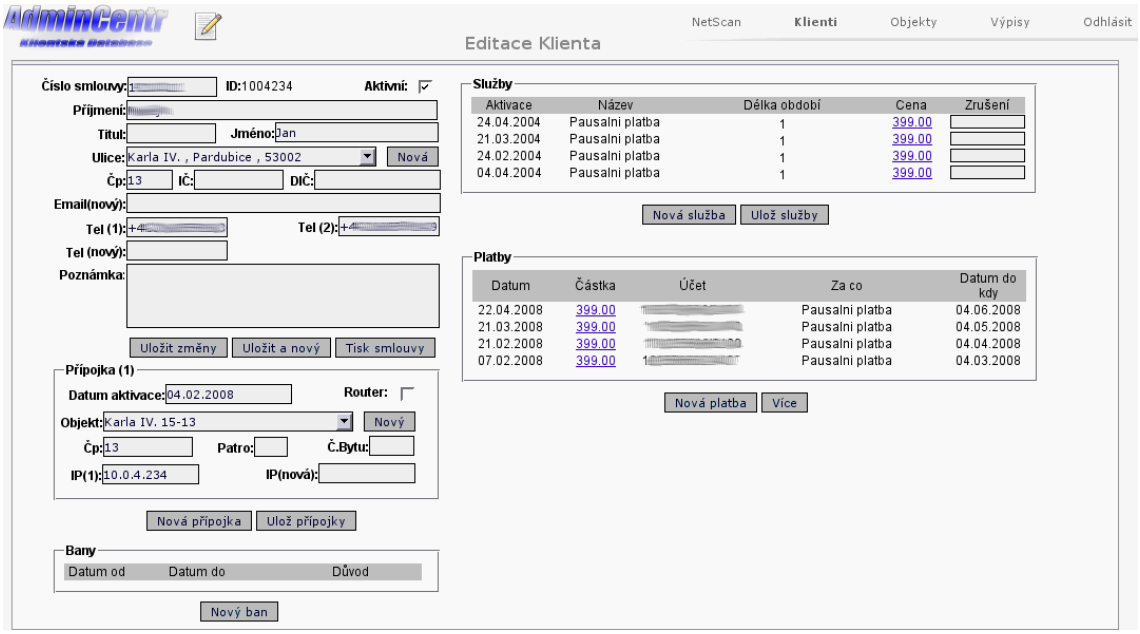
Obr. 3.9: Webová stránka kartotéky klientů

Celé zobrazení je realizováno pomocí kaskádových stylů CSS a jednoduchých HTML objektů. Toto uspořádání umožňuje snadnou změnu vzhledu bez nutnosti zásahu do obslužných skriptů PHP a to pouhou úpravou kaskádového stylu. Navíc je tímto řešením ve spojení s asynchronní komunikací se serverem pomocí javascriptu sníženo množství přenášených dat. Listování mezi stránkami, řazení položek a změna filtrace jsou totiž opět realizovány pomocí AJAXu tak, aby byla obnovována pouze tabulka položek, nikoli celá stránka. Obslužný PHP script pak v tomto případě realizuje pouze SQL dotaz do databáze a jeho výstup odešle klientské aplikaci. Na straně klienta je poté javascriptem zobrazen požadovaný seznam.

Pomocí AJAXu jsou také získávány nápovědné informace pro zadávání filtrace zobrazení, tak aby bylo možné předcházet chybám ze strany uživatele systému.

Podrobnosti klienta

V zobrazení podrobností klienta je již možné měnit jednotlivé hodnoty polí. Na obrázku 3.10 je vidět vzhled tohoto zobrazení. Kromě běžných údajů jsou zde zobrazeny poslední platby za služby, přidělené IP adresy atd. Celé zobrazení je rozděleno na oblasti. První oblast obsahuje obecné údaje o klientovi. Druhá seznam přípojek, které je možné editovat. Třetí oblast je zaměřena na platby. Upravovatelné položky



Obr. 3.10: Webová stránka editace klienta

jsou již při zadávání pomocí javascriptu kontrolovány a přislušné chybné pole je zvýrazněno, aby bylo opět předcházeno chybám na straně uživatele.

Seznam objektů

Vzhledem k výše zmiňované obvyklé topologii sítě jsou klienti sdružováni do objektů. Každý objekt má vlastní podsíť případně podsítě IP adres přidělovaných klientům (respektive přípojkám, podrobněji v kapitole 3.4). S objekty je poté pracováno podobně jako s klienty. Je implementováno zobrazení seznamu objektů a rozhraní pro jejich editaci.

Výpisy z účtů

Neméně důležitou částí klientské databáze je zobrazení výpisů z účtů. Jsou zde zobrazeny jednotlivé položky příslušných výpisu z účtů. Výpisem z účtu je v tomto případě také pokladní výpis, kde jsou zaznamenány všechny zadané položky placené hotovostně. Pokladní výpis je vždy sdružován po čtvrtletích. Ostatní výpisy jsou za období dle importu do databáze. V současné době je vytvořen import pro výpis z eBanky a ČSOB. Import je prováděn pomocí formuláře, jehož parametrem je soubor s elektronickým výpisem z banky. U výpisu z ČSOB je předpokládán formát „skl“. U eBanky je předpokládán formát HTML. Importovaný výpis je nejprve nahrán na server a poté pomocí skriptu napsaného v pythonu zpracován a případně uložen do databáze.

Platby od uživatelů jsou identifikovány na základě variabilního symbolu, který by měl odpovídat číslu smlouvy s klientem. Platby pro něž nebyla nalezena příslušnost ke klientovi jsou označeny jako nespárované a je možné je ručně přiřadit.

3.2.3 Zabezpečení

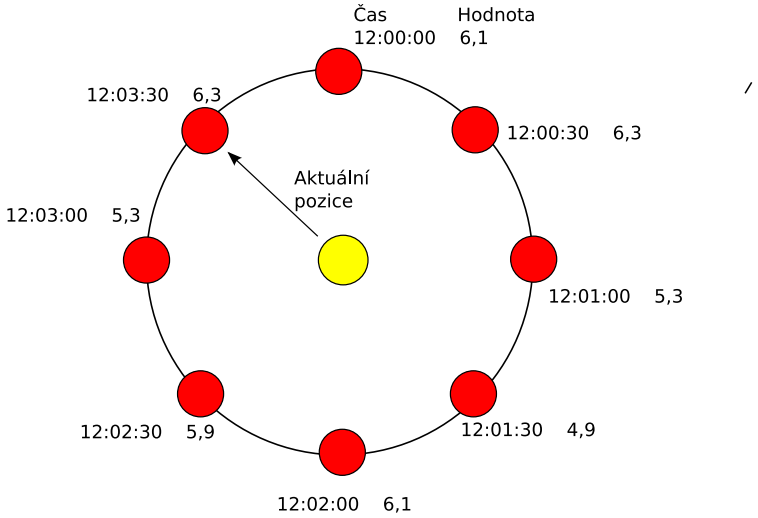
K datům dohledového systému by měl mít pouze oprávněný uživatel a hesla uživatelů by měla být chráněna proti odposlechu dat. Tohoto je u systému docíleno použitím zabezpečení pomocí SSL na straně serveru. Je proto nutné, aby webový server toto zabezpečení umožňoval. Zpravidla je toto spojení provozováno na portu 443 narozdíl od standardního webového portu 80. Touto cestou je možné zamezit odposlechu chráněných dat a tím i přístupových hesel. Přístup na zabezpečené vrstvě k jednotlivým částem webového rozhraní je poté chráněn heslem každého uživatele. Podle typu uživatele, který je definován v tabulce uživatelů v databázi (viz. 3.4), je umožněn přístup k různým možnostem systému. V podstatě se jedná především o dvě skupiny uživatelů a to uživatelé s právem zápisu a s právem pouze pro čtení. Práva čtení a zápisu jsou nastavována zvlášť pro jednotlivé části systému. Například pro potřeby učetnictví není nutný přístup s právem zápisu do dohledové části systému.

3.3 Databáze RRD

Vzhledem k množství dat, která jsou získána v každém testovacím cyklu, není vhodné použití klasické SQL databáze, neboť ani není nutné, aby v dlouhodobém horizontu byla uchovávána všechna. Pro potřebu dlouhodobých přehledů postačí pouze průměrované hodnoty. Pro tyto účely byla speciálně navržena tzv. RRD (Round Robin Database), tato databáze je součástí tzv. RRDTool, což je projekt sdružující

vlastní databázi a také podporu pro její úpravy a vytváření grafů z těchto databází. Autorem projektu je Tobias Oetiker a více o tomto projektu je možné nalézt na stránkách projektu [3].

V případě RRD se jedná o lineární databázi. U této databáze se nová hodnota vždy zapisuje navrch tabulky. Na obrázku 3.11 je vidět celý princip ukládání hodnot do databáze. Z uvedeného vyplývá, že je-li překročena délka databáze jsou automaticky přepisována data od začátku a samozřejmě se také s každým zápisem mění také ukazatel na počátek databáze. Význam určeného kroku databáze je vidět také na obrázku 3.12. V daném časovém úseku je vytvořen pouze jediný primární datový bod a to jako průměr hodnot zapisovaných do databáze. Pokud v časovém kroku není do databáze vkládána žádná hodnota je bodu nastavena předdefinovaná hodnota nebo neznámá hodnota.



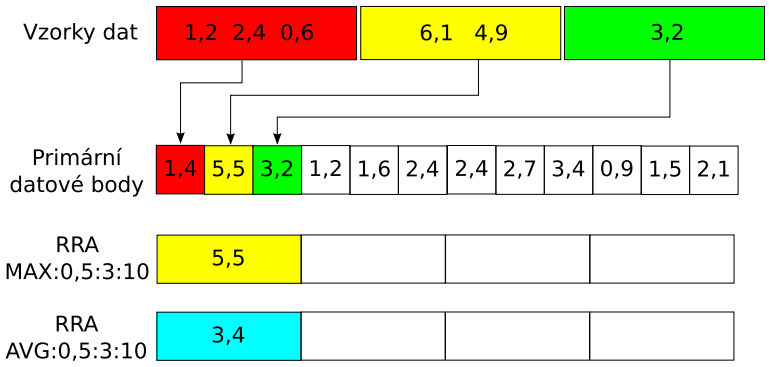
Obr. 3.11: Princip Round-Robin databáze

Při vytváření databáze je jasně definován minimální krok pro zápis do databáze, velikost hodnot zapisovaných do databáze, počet hodnot a také délka databáze. Tímto je dáno, že výsledný databázový soubor má vždy konstantní velikost a nezatěžuje tak souborový systém v závislosti na zapsaných datech. Pro uchování dat v delším časovém horizontu jsou definovány archivy RRA (Round Robin Archive). Tyto archivy uchovávají dle nastavení data do tabulky o definované velikosti. Data jsou do archivu vkládána jako průměrná, minimální, maximální nebo poslední hodnota za definovaný interval. Princip získávání hodnot z tabulky primárních datových bodů je závislý na parametrech RRA a jeho principy jsou zřetelné na obrázku 3.12. Pomocí archívů je možné zachovávat přehledová data za měsíce případně roky. Počet archívů není omezen a navíc je dle potřeby možné RRA přidávat i do již vytvořené databáze. RRDTool navíc implementují ideální nástroj pro vytváření grafů z dat

uložených v RRD, což je jedním z hlavních faktorů, jež ovlivnili volbu pro použití v dohledovém systému.

Tab. 3.2: Přehled RRA vytvářených v základním nastavení

Optimalizace pro období	Konsolidační funkce	Náhrada neznámých	Počet vstupních bodů	Počet bodů RRA	Parametr RRDTool
8 hodin	Průměr	0,5	1	500	RRA:AVERAGE:0.5:1:500
den (24 hodin)	Průměr	0,5	3	500	RRA:AVERAGE:0.5:3:500
týden (120 hodin)	Průměr	0,5	15	500	RRA:AVERAGE:0.5:15:750
měsíc (720 hodin)	Průměr	0,5	90	500	RRA:AVERAGE:0.5:90:500
rok (720 hodin)	Průměr	0,5	1200	500	RRA:AVERAGE:0.5:1200:450



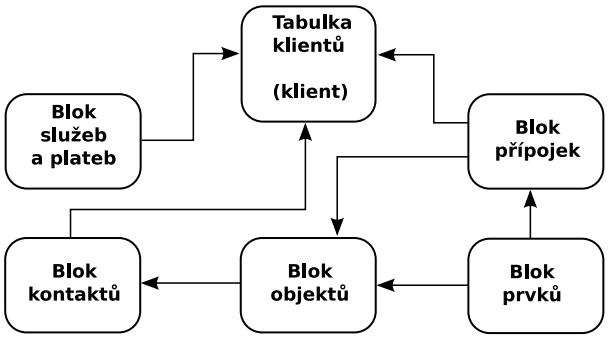
Obr. 3.12: Příklad vzniku primárních datových bodu a RRA

Pomocí RRDTool jsou uchovávána data o odezvách jednotlivých prvků v síti, ztrátovost, dostupnost nadřazeného prvku, počet odeslaných a přijatých paketů v jednom testu . Pro přehlednost a možnost manipulace jsou pro každý prvek sítě definovaný v tabulce bodů vytvořeny dvě RRD. První obsahuje vždy minimální, maximální a průměrnou odezvu získanou z jednoho měření. Druhá databáze uchovává počet odeslaných a přijatých paketů, vypočtenou ztrátovost paketů a informaci zda nadřazený bod vykazuje závadu. Poslední údaj je uchováván, aby bylo možné lokalizovat závadu, neboť ztrátovosti jednotlivých linek, pokud jsou za sebou, se sčítají. V základním nastavení mají databáze definovány RRA pro uchování dat na délku až jednoho roku. Archivů je v základním nastavení definováno pět. Tyto archivy jsou nastaveny na vytváření grafů pro různé časové úseky, podrobněji jsou parametry jednotlivých archívů v tabulce č. 3.2. Samozřejmě lze vytvářet grafy i pro jiné délky období, avšak již s nutností přepočtů atd. Jak již bylo řečeno je však možné přidat i RRA pro uchování dat i na delší období apod.

Přístup k RRD není nikterak uzpůsoben pro moduly a je tedy nutné zápis a vytvoření databází v modulu zajistit.

3.4 Databáze PostgreSQL

Pro uchovávání dat jako jsou IP adresy síťových prvků, seznamy uživatelů apod. je použita databáze PostgreSQL. PostgreSQL je plnohodnotným relačním databázovým systémem s otevřeným zdrojovým kódem. Má za sebou více než patnáct let aktivního vývoje a má vynikající pověst pro svou spolehlivost a bezpečnost. Databáze je portována do všech běžně rozšířených operačních systémů včetně Linuxu, UNIXů a Windows. Stoprocentně splňuje podmínky ACID (automatizace, konzistence, izolace, trvanlivost – Atomicity, Consistency, Isolation, Durability), plně podporuje cizí klíče, operace JOIN, pohledy, spouště a uložené procedury. Obsahuje většinu SQL92 a SQL99 datových typů, např. INTEGER, NUMERIC, BOOLEAN, CHAR, VARCHAR, DATE, INTERVAL a TIMESTAMP. K PostgreSQL existuje navíc kvalitní volně dostupná dokumentace [4].



Obr. 3.13: Blokové zobrazení částí databáze

Tab. 3.3: Popis polí tabulky uživatelů a klientů

Název tabulky	Název pole	Typ pole	Délka	Popis
klient	csml	varchar	20	Číslo smlouvy
	titul	varchar	8	Titul
	prijmeni	varchar	40	Příjmení/Název firmy
	jmeno	varchar	20	Křestní jméno
	ic	varchar	8	IČ
	dic	varchar	10	DIČ
	ul_id	integer	-	Vazba na tabulku ulic
	cp	integer	-	Číslo popisné
	isactive	boolean	-	Příznak odstranění
	pozn	text	-	Poznámka
	klient_id	serial	-	Primární klíč
users	usr_id	integer	-	Primární klíč
	usr_name	varchar	20	Název uživatele
	usr_active	boolean	-	Platnost uživatele
	usr_time_od	smallint	-	Čas pro zahajení doručování zpráv
	usr_time_do	smallint	-	Čas pro ukončení doručování zpráv
	usr_week_day	bit	7	Výběr dní v týdnu pro doručování zpráv
	usr_type	bit	8	Typ doručovaných zpráv
	usr_passwd	character	32	Heslo uživatele
	usr_prava	bit	8	Přístupová práva uživatele

Vzhledem k propojení dat ukládaných do databáze byl nutný vhodný návrh celé databáze. Data jsou rozdělena na několik základních tabulek, jak je vidět na

obrázku celé struktury databáze v příloze A, kde jsou navíc zřetelné závislosti mezi tabulkami a podtržením zvýrazněny primární klíče tabulek. Podrobnější informace o jednotlivých položkách jsou poté uspořádány v tabulkách 3.3, 3.5, 3.6, 3.7, 3.8 a 3.9. Celé toto výsledné uspořádání vychází ze zásad pro tvorbu datových struktur databáze [5].

Vlastní model databáze je možné rozdělit na několik částí tak, jak jsou, mimo tabulky uživatelů, zobrazeny na obrázku 3.13. Hlavním spojovacím prvkem jednotlivých částí je tabulka klientů nesoucí název „klient“. Tato tabulka má jako primární klíč položku „klient_id“. Jedná se o položku typu SERIAL, což je datový typ odpovídající typu INTEGER, ale narozdíl od něho je určen k tvorbě primárních klíčů, neboť jeho výchozí hodnotou je následující hodnota sekvence. Proto není možné, aby došlo ke zdvojení primárního klíče a tím k chybě. Na položku „klient_id“ se poté odkazují další tabulky a případně také tabulky modulů. Význam ostatních položek tabulky klientů je uveden v tabulce 3.3.

Tab. 3.4: Popis bitů oprávnění

Bit č.	**Význam**
0	Oprávnění čtení z databáze klientů
1	Oprávnění zápisu do databáze klientů
2	Oprávnění čtení z dat dohledového systému
3	Oprávnění zápisu nastavení prvků
4	Oprávnění změn v platbách
5	Oprávnění přidávat uživatele systému
6	Nevyužito
7	Nevyužito

Pro účely přístupu a zajištění bezpečnosti je v databázi tabulka uživatelů s názvem „users“, která obsahuje jednotlivá pole tak jak jsou uvedeny v tabulce 3.3. Pro přihlašování do systému jsou nejdůležitější položky „usr_name“ a „usr_passwd“, ve kterých je uloženo přihlašovací jméno uživatele a heslo pro přihlášení v podobě MD5 HASHe. Heslo je kódováno, aby nebylo možné jeho zneužití v případě vniknutí útočníkem do databáze. Po vstupu do webové aplikace má každý uživatel nastavena oprávnění dle položky „usr_prava“. Jedná se o kombinaci osmi bitů. Každý bit určuje oprávnění dle tabulky 3.4. Za povšimnutí stojí také položky „usr_time_od“, „usr_time_do“ a „usr_week_day“. Tyto položky určují čas a den v týdnu, kdy je uživatel upozorňován výstražnými zprávami. Položky „usr_time_od“ a „usr_time_do“ určují hodinu od/do kdy jsou zprávy doručovány v aktuálním dni. Položka „usr_week_day“ je 7-bitová binární položka, kde každý bit odpovídá dni v týdnu bráno od pondělí. Hodnota 1 určuje, že uživatel bude v daném dni upozorňován a hodnota 0 naopak upozorňování deaktivuje pro příslušný den v týdnu.

Položka „usr_type“ je také svázána s informačními zprávami a určuje masku typu zpráv zasílaných uživateli.

3.4.1 Blok služeb a plateb

Prvním blokem je blok služeb a plateb. V tomto bloku databáze jsou uchovávána data o službách klientům poskytovaných a také o platbách za tyto služby. Navíc je zde vložena tabulka s názvem „ban“, která obsahuje seznam přerušení jednotlivých služeb. Na obrázku 3.14 je vidět podrobněji struktura tabulek včetně jejich propojení. Popis jednotlivých položek tabulek jsou poté přehledně v tabulce 3.5

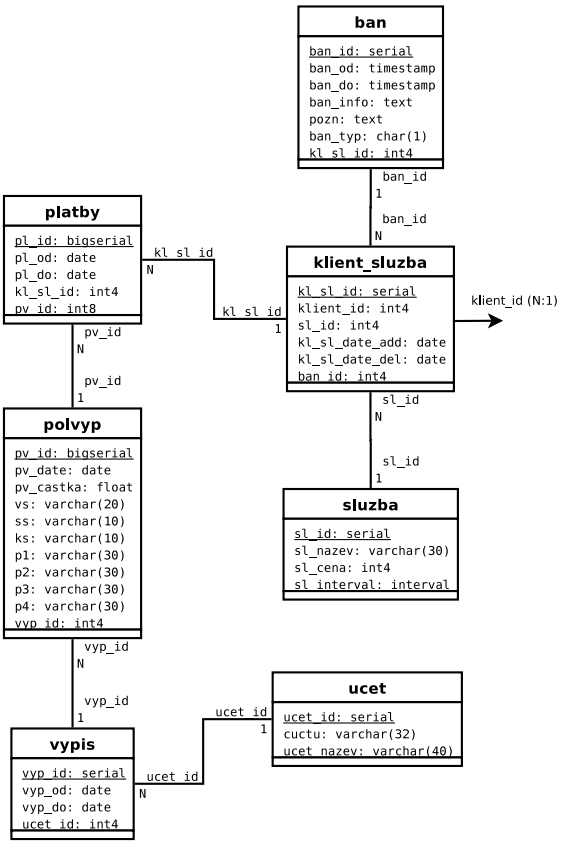
Tabulka „ban“ je určena především pro řešení firewallu poskytovatele, aby bylo možné omezit připojení klientů do sítě Internet a případně je upozornit na nesrovnalosti, kvůli kterým byli omezeni.

Platby jsou jak již bylo zmíněno dříve zaznamenány pomocí struktury výpisů. Výpis přísluší ke konkrétnímu účtu poskytovatele definovaného v tabulce s názvem „ucet“. Každý výpis obsahuje příslušné položky reprezentované položkami v tabulce s názvem „pol_vyp“ na obrázku 3.14. Příslušnost konkrétní platby (resp. položky výpisu) ke službě je realizována skrze tabulku s názvem „platby“. Tato tabulka umožňuje, aby jedna položka výpisu byla úhradou vícero služeb a zároveň tabulka mohla obsahovat více úhrad za jednotlivá období. Tabulka s názvem „klient_sluzba“ je tabulkou, jež realizuje propojení plateb, „banů“ a služeb s tabulkou klientů. Mezi tabulkou služeb a klientů je relace M:N, kdy více uživatelů může využívat jeden typ služby a naopak jeden klient může využívat více služeb. Proto tato tabulka provádí řešení tohoto relačního vztahu. Navíc obsahuje datumy přidání (případně odebrání) služby tak, aby bylo možné provádět kontrolu plateb za službu zpětně.

Pro účely přidávání položek výpisů (např. při importování výpisů z bankovních účtů) je v databázi vytvořena vnitřní funkce porovnávající automaticky variabilní symbol položky s čísly smluv. Jedná se o takzvaný „trigger“, který je spuštěn před přidáním položky do tabulky. Během zpracování je zjištěno zda platba dle variabilního symbolu přísluší konkrétnímu klientovi nebo službě a realizuje přidání položky do tabulky plateb. Pokud není nalezena příslušnost položky výpisu je ponechána ve stavu nespárování a je nutné ji ručně přiřadit pomocí webového rozhraní. Celou tuto operaci by bylo možné realizovat ve webové aplikaci, avšak realizace uvnitř databáze dociluje vyššího výkonu a lepší odezvy importu dat s mnoha položkami.

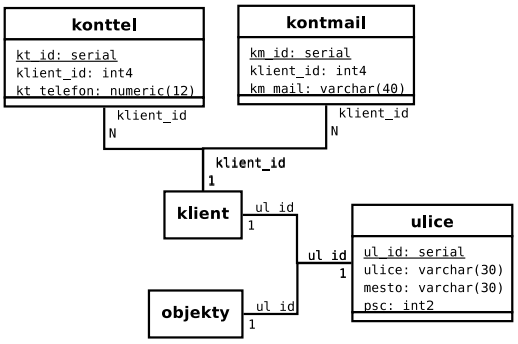
3.4.2 Blok kontaktů

Dalším blokem databáze je blok kontaktů, který je podrobněji zobrazen na obrázku 3.15. Blok obsahuje tabulku kontaktních emailových adres, tabulku kontakt-



Obr. 3.14: Tabulky bloku služeb a plateb

ních telefonů a kontaktních adres (resp. ulic včetně města a čísla popisného). Toto uspořádání umožňuje každému uživateli přiřadit libovolný počet kontaktních telefonů a emailů. Tabulka s názvem „ulice“ je společná také pro blok přípojek databáze a je zavedena především pro zmenšení velikosti celé databáze, neboť tato data jsou u mnohých klientů stejná. Navíc se dá pomocí této tabulky usnadnit zadávaní nových klientů do databáze ve webové aplikaci.



Obr. 3.15: Tabulky bloku kontaktů

Tab. 3.5: Popis polí tabulek bloku služeb a plateb

Název tabulky	Název pole	Typ pole	Délka	Popis
ban	ban_id	serial	-	Primární klíč
	ban_od	timestamp	-	Počátek omezení
	ban_do	timestamp	-	Konec omezení
	ban_info	text	-	Informace pro zákazníka
	pozn	text	-	Poznámka správce
	ban_typ	character	1	Typ V/S/P/J (virus/spam/platby/jiné)
	kl_sl_id	integer	-	Vazba na tabulku klient_sluzba
klient_sluzba	kl_sl_id	serial	-	Primární klíč
	klient_id	integer	-	Vazba na tabulku klientů
	sl_id	integer	-	Vazba na tabulku služeb
	kl_sl_date_add	date	-	Datum přidání služby uživateli
	kl_sl_date_del	date	-	Datum odebrání služby
	ban_id	integer	-	Vazba na tabulku ban
platby	pl_id	bigserial	-	Primární klíč
	pl_od	date	-	Počátek placeného období
	pl_do	date	-	Konec placeného období
	kl_sl_id	integer	-	Vazba na tabulku klient_sluzba
	pv_id	bigint	-	Vazba na tabulku položek výpisu
polvyp	pv_id	bigserial	-	Primární klíč
	pv_date	date	-	Datum placení
	pv_castka	double	-	Placená částka
	vs	varchar	20	Variabilní symbol
	ss	varchar	10	Specifický symbol
	ks	varchar	10	Konstantní symbol
	p1	varchar	30	Další pole výpisu
	p2	varchar	30	Další pole výpisu
	p3	varchar	30	Další pole výpisu
	p4	varchar	30	Další pole výpisu
	vyp_id	integer	-	Vazba na tabulku výpisů
sluzba	sl_id	serial	-	Primární klíč
	sl_nazev	varchar	30	Název služby
	sl_cena	integer	-	Cena služby
	sl_interval	interval	-	Interval platby za službu
ucet	ucet_id	integer	-	Primární klíč
	cuctu	varchar	32	Číslo účtu
	ucet_nazev	varchar	40	Název účtu
vypis	vyp_id	serial	-	Primární klíč
	vyp_od	date	-	Počátek období výpisu
	vyp_do	date	-	Konec období výpisu
	ucet_id	integer	-	Vazba na tabulku účtů

3.4.3 Blok přípojek

Vzhledem ke skutečnosti že jeden klient může mít více přípojek od poskytovatele, je nutné pro tuto skutečnost databázi přizpůsobit. Tento požadavek je realizován blokem přípojek. Pro klienta je možné přidat libovolný počet přípojek. Každá přípojka je navíc lokalizována propojením s blokem objektů.

Přípojce je možné přiřadit opět libovolný počet IP adres, které jsou uloženy v tabulce s názvem „ip“ (jak je vidět na schématu databáze v příloze A). Tato

Tab. 3.6: Popis polí tabulek kontaktů

Název tabulky	Název pole	Typ pole	Délka	Popis
kontmail	km_id	serial	-	Primární klíč
	klient_id	integer	-	Vazba na tabulku klientů
	km_mail	varchar	40	Emailová adresa
konttel	kt_id	serial	-	Primární klíč
	klient_id	integer	-	Vazba na tabulku klientů
	kt_telefon	numeric	12	Telefonní číslo
ulice	ul_id	integer	-	Primární klíč
	ulice	varchar	30	Název ulice
	mesto	varchar	30	Město
	psc	smallint	-	Poštovní směrovací číslo

Tab. 3.7: Popis polí tabulek bloku přípojek

Název tabulky	Název pole	Typ pole	Délka	Popis
ip	ip_id	serial	-	Primární klíč
	ip_ip	inet	-	IP adresa (adresa sítě s prefixem)
	obj_id	integer	-	Vazba na tabulku objeků
	prip_id	integer	-	Vazba na tabulku přípojek
	prv_id	integer	-	Vazba na tabulku prvků
mac	mac_id	serial	-	Primární klíč
	mac	macaddr	-	MAC adresa
	vyrobce	varchar	30	Nalezený výrobce zařízení dle MAC
	potvrzena	boolean	-	Potvrzení platnosti MAC adresy
	ip_id	integer	-	Vazba na tabulku IP adres
pripojka	klient_id	integer	-	Vazba na tabulku klientů
	obj_id	integer	-	Vazba na tabulku objektů
	cp	integer	-	Číslo popisné umístění přípojky
	date_add	date	-	Datum přidání přípojky
	date_del	date	-	Datum odebrání přípojky
	prip_patro	smallint	-	Patro přípojky
	prip_cbytu	varchar	5	Číslo bytu
	prip_id	serial	-	Primární klíč

tabulka obsahuje také IP adresy prvků sítě, které jsou obsaženy v bloku prvků databáze. Aby bylo možné tabulku realizovat skládá se položka tabulky z vlastní IP adresy (případně adresy sítě), a třech identifikátorů odkazujících na tabulku přípojek, objektů a prvků, tak jak je vidět v tabulce 3.7. Podle příslušnosti IP adresy je možné, aby příslušný identifikátor nabýval hodnoty „NULL“. Dle hodnot „NULL“ je poté možné rychle rozlišit položky příslušící k tabulce prvků, přípojek či pouze objektu.

K IP adresám jsou přiřazovány MAC adresy. Klient tedy může k připojení na jedné IP adrese využívat vícero zařízení s různými MAC adresami. Tabluku MAC adres je navíc možné doplňovat modulem SNMP systému z data získaných ze síťových prvků. Vzhledem k faktu, že MAC adresy jsou mezinárodně přidělovány výrobcům síťových zařízení, je možné MAC adresy rozlišovat dle výrobce.

3.4.4 Blok objektů

Blok objektů je složen v zásadě pouze tabulkou objektů včetně položek uvede-

Tab. 3.8: Popis polí tabulky objektů

Název tabulky	Název pole	Typ pole	Délka	Popis
objekt	obj_id	serial	-	Primární klíč
	ul_id	integer	-	Vazba na tabulku ulic
	cp_od	integer	-	Počáteční číslo popisné
	cp_do	integer	-	Koncové číslo popisné
	nazev	varchar	20	Název objektu
	pozn	text	-	Poznámka k objektu

ných v tabulce 3.8. Jedná se o objekty, ve kterých poskytovatel provozuje služby či má umístěné prvky. Především je blok objektů zaveden pro větší přehlednost a také pro snazší získávání statistických údajů, které napomáhají při návrhu páteřní sítě. Prostřednictvím této tabulky je možné efektivně zjišťovat počet klientů, prvků či přípojek v daném objektu případně lokalitě. Tato data by bylo sice možné získávat i složitějším zpracováním ostatních tabulek, ale použité řešení napomáhá navíc charakterizovat jednotlivé objektu (např. počet vchodů).

S tabulkou objektů je svázána tabulka IP adres, jak bylo již řešeno výše. K objektu se mohou vázat nejenom IP adresy, ale také adresy sítí. Adresy náležící pouze k objektu, nikoli k přípojce, mají v prázdnou hodnotu pole „prip_id“.

3.4.5 Blok síťových prvků

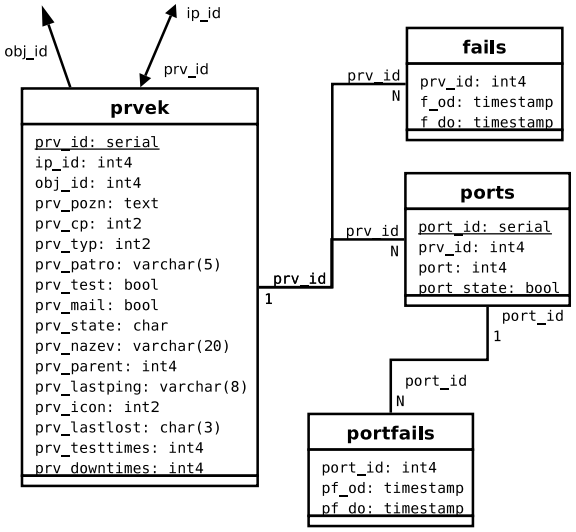
Blok prvků je hlavním blokem pro samotnou testovací aplikaci a celou dohledovou část systému. Blok obsahuje především tabulku prvků. I v této tabulce mají prvky jedinečný identifikátor (primární klíč), pomocí něhož se ostatní tabulky váží, jak je vidět na obrázku 3.16, kde jsou zobrazeny všechny tabulky bloku prvků včetně jejich propojení. Je zde také vidět, že prvky mohou mít také přiřazeno více IP adres z tabulky „ip“. Pouze jedna IP adresa je však výchozí a je tedy v testovací aplikaci využívána např. pro ICMP test. Význam ostatních polí tabulek je podrobněji zaznamenán v tabulce 3.9. Tabulka prvků obsahuje také příznak „prv_test“ určující, zda má být prvek podrobován ICMP testu, neboť pro některé síťové prvky (např. některé switche) nemohou být testovány ICMP testem. Pro potřeby zobrazení nejaktuálnějších naměřených hodnot jsou do tabulky prvků zaznamenávány poslední naměřené hodnoty odezvy a ztrátovosti. Tato data by sice bylo možné získávat z RRD, přístup do RRD je však náročnější a je proto pro potřeby zobrazení posledních hodnot eliminován.

Pro zaznamenávání závad prvků zjištěných ICMP testem je v databázi tabulka s názvem „fails“, kde je uveden čas počátku a konce poruchy a její typ. Každá porucha náleží k příslušnému bodu a proto je nutné, aby součástí tabulky byl odkaz na příslušný prvek, k němuž závada náleží (viz. tab. 3.9).

Tab. 3.9: Popis polí tabulek bloku síťových prvku

Název tabulky	Název pole	Typ pole	Délka	Popis
fails	prv_id	integer	-	Vazba na tabulku prvků
	f_od	timestamp	-	Počátek poruchy
	f_do	timestamp	-	Konec poruchy
portfails	port_id	integer	-	Vazba na tabulku portů
	pf_od	timestamp	-	Počátek výpadku portu
	pf_do	timestamp	-	Konec výpadku portu
ports	port_id	integer	-	Primární klíč
	prv_id	integer	-	Vazba na tabulku prvků
	port	integer	-	Číslo TCP portu
	port_state	boolean	-	Aktuální stav portu
prvek	prv_id	serial	-	Primární klíč
	ip_id	integer	-	Vazba na tabulku IP adres
	obj_id	integer	-	Vazba na tabulku objektů
	prv_pozn	text	-	Poznámka k prvku
	prv_cp	smallint	-	Číslo popisné
	prv_typ	smallint	-	Typ prvku
	prv_patro	varchar	5	Patro umístění prvku
	prv_test	boolean	-	Příznak testování prvku
	prv_mail	boolean	-	Příznak upozornění správce o výpadku
	prv_maxping	float	-	Maximální přípustná hodnota odezvy
	prv_maxlost	integer	-	Maximální přípustná hodnota ztrátovosti v procentech
	prv_state	character	1	Aktuální stav prvku
	prv_nazev	varchar	20	Název prvku
	prv_parent	integer	-	Identifikátor nadřazeného prvku
	prv_lastping	varchar	8	Poslední změřená průměrná odezva
	prv_icon	smallint	-	Číslo ikony ve stromu prvků
	prv_lastlost	character	3	Poslední změřená ztrátovost
	prv_testtimes	integer	-	Počet testování
	prv_downtimes	integer	-	Počet testování s chybou

K prvku je možné také přiřadit libovolný počet testovaných portů. Tyto porty jsou uloženy v tabulce s názvem „ports“ včetně jejich aktuálního stavu zjištěného testovací aplikací případně nastaveným z webového rozraní. Pro zaznamenávání výpadků portů je poté určena tabulka „portfails“ obsahující počátek a konec výpadků s ukazatelem na příslušný port.



Obr. 3.16: Tabulky bloku síťových prvků

4 INSTALACE A NASTAVENÍ APLIKACE

4.1 Instalace

Pro použití celé aplikace je nutné nejprve aplikaci nainstalovat a splnit závislosti pro její spuštění. Požadavky systému jsou :

- Python 2.5,
- modul psycopg2 pro Python,
- modul pysnmp pro Python,
- modul rrdtool pro Python,
- RRDTool verze 1.23 a vyšší,
- databázový server PostgreSQL 8.3,
- webový server Apache (doporučeno 2.2.8 a vyšší),
- mod_python pro Apache,
- mod_ssl pro Apache,
- mod_php pro Apache (PHP 5.2 a vyšší).

Při splnění těchto závislostí je možné nakopírovat soubory do příslušné složky, kde chceme aplikaci umístit. Dále je nutné nainstalovat databázi a to spuštěním souboru databáze.sql pomocí příkazového interpretu PostgreSQL databáze. Pro přídavné moduly je nutné spustit jejich databázové soubory.

Po instalaci databáze je nutné nastavit parametry aplikace pomocí souboru `NetScan.conf` umístěného v podadresáři `etc` aplikace. Nastavení tohoto souboru je popsáno dále v kapitole 4.2. Pokud je aplikace nastavena je možné spustit testovací aplikaci. Ta nebude provádět žádné testy neboť nejsou nastaveny žádné prvky sítě.

Dalším krokem je nastavení webserveru Apache aby přistupoval k adresáři webových stránek. Pokud je webserver správně nastaven je možné započít práci se systémem. Výchozí uživatel je „admin“ a heslo je „admin1234“.

4.2 Nastavení NetScan.conf

Nastavení aplikace je, jak bylo již řečeno, realizováno pomocí souboru NetScan.conf v podadresáři `etc` aplikace. Soubor obsahuje seznam proměnných nastavených uživatelem. Hodnota proměnných je zadávána obdobně jako v python. Nejprve je zadán název proměnné a za rovnítko je zadána hodnota této proměnné. Pokud proměnná není v konfiguračním souboru zadána je použita výchozí hodnota aplikace. Seznam možných použitých proměnných je uveden v tabulce 4.1. Je nutné dodržovat syntaxy zadávání proměnných tak jak je uvedena ve sloupci příkladů tabulky.

Tab. 4.1: Popis možných nastavení NetScan.conf

Název proměnné	**Popis**	**Výchozí hodnota**	**Příklad**
MaxPingThreads	Maximální počet vláken Ping	5	MaxPingThreads = 10
CycleTime	Délka základního cyklu hlavního programu v sekundách	60	CycleTime=120
RrdDir	Cesta pro ukládání RRD	'/usr/local/NetScan/rrd'	RrdDir='/RRD/NetScan'
CreateDirs	Vytvoř neexistující adresáře	True	CreateDirs=False
timeToDeath	Počet návazných nesplnění kritérií, po kterých je prvek prohlášen za vadný	10	timeToDeath=10
Fork	Příznak běhu na pozadí (pouze Linux)	True	Fork=False
DatabaseName	Název SQL databáze	'NetScan'	DatabaseName='test'
DatabaseHost	Host pro připojení k databázi	'127.0.0.1'	DatabaseHost='192.168.1.1'
DatabasePort	Číslo portu na němž naslouchá PostgreSQL	5432	DatabasePort=5433
DatabaseUser	Uživatel pro připojení k databázi	'NetScan'	DatabaseUser='pokus'
DatabasePass	Heslo pro připojení k databázi	''	DatabasePass='neco'
DatabaseSSL	Nastavení zabezpečení připojení k databázi	'allow'	DatabaseSSL='required'
DatabaseCp	Klientská kódová stránka pro databázi	'UTF-8'	DatabaseCp='ISO-8889-2'
Modules	Pole názvů spouštěných modulů	[]	Modules=['SNMP','TCPD']
SmtpHost	Host pro odesílání emailů	'127.0.0.1'	SmtpHost='mail.nic.cz'
SmtpPort	Port pro odesílání emailů	25	SmtpPort=2525

5 NASAZENÍ A TESTY APLIKACE

V současné době je aplikace instalována a testována na síti poskytovatele, který zajišťuje připojení k internetu pro přibližně 2500 klientů. V systému registrováno než 250 síťových prvků. Tyto prvky jsou testovány ICMP testem a o vybraných prvcích (přibližně 100) jsou zjišťovány další informace pomocí SNMP protokolu. Jedná se především o údaje o množství přenesených dat, vytížení procesoru a MAC adresy připojených uživatelů. Během dvouměsíčního testování nebyly odhaleny závažné vady a to ani v simulovaných krizových situacích, kdy byl server záměrně odpojen od sítě. V tomto případě probíhají jednotlivé testy nejdelší možnou dobu, poté vzniká možnost přeplňování fronty spuštění testů. Pokusem byl zjištěno, že při základním nastavení, kdy je délka cyklu systému je 60 vteřin, nedochází k potížím ani při 500 prvcích sítě k otestování. Jediným nutný uzpůsobením bylo přidání funkce odstraňující diakritiku z emailových zpráv, neboť v případě přesměrování zpráv na mobilní telefon diakritika způsobovala obtíže.

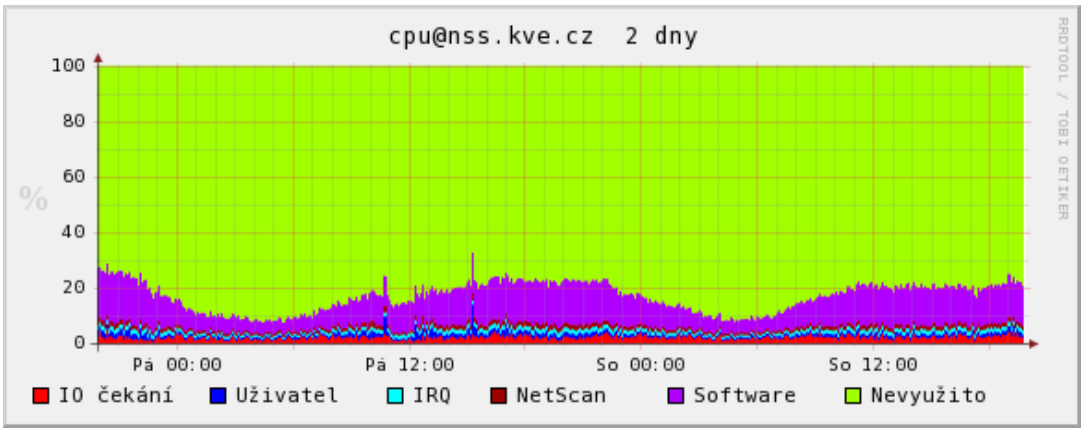
Tab. 5.1: Parametry testovacího serveru

Procesor	INTEL Quad-Core XEON X3210 4x 4270 bogomips
Platforma	INTEL Server Aspen Hill (SR1530AH)
Paměť	4 GB DDR2
Operační systém	Linux
Služby	Mail, PostgreSQL, SSH, WWW (Apache + mod_php + OpenSSL)
Verze Pythonu	2.5
Verze PHP	5.2.5

Sledováním bylo také zjišťováno vytížení procesoru dohledovým systémem. Analýza byla prováděna na serveru jehož parametry jsou uvedeny v tabulce 5. Mailserver obsluhoval poštu pouze vytvářenou Dohledovým systémem. Webový server obsluhoval pouze 10 klientů, kteří obnovovali pravidelně webové stránky realizované aplikace. Databázový server PostgreSQL sloužil také pouze pro účely aplikace. K serveru byl připojen jeden uživatel pomocí SSH (Secure Shell). Celkové vytížení jediného využívaného jádra procesoru i při různých extrémních situacích nepřesáhlo 5 procenta. Vytížení procesoru samotným jádrem dohledového systému nepřesáhlo 2 procenta. Největší zatížení představovalo především zapisování do RRD databází a to v podobě přístupů k pevným disků.

Po dobu dvou měsíců, jak bylo řečeno, byla aplikace provozována na serveru se stejnou konfigurací, jaká je uvedena v tabulce 5. Server obsluhoval datové toky okolo 100 Mbps (20 kpps) a realizoval mail server včetně spam filtru pro všechny

uživatele poskytovatele. Dále server realizoval veřejný webový server poskytovatele a některých klientů. Průběh vytížení procesoru serveru za období dvou dní je vidět na obrázku 5.1. Z grafu je zřetelné, že vytížení dohledovým systémem nepřekročilo vytížení v řádu jednotek procent.



Obr. 5.1: Graf vytížení CPU serveru v reálné aplikaci

6 ZÁVĚR

Diplomová práce se zabývala rozborem sítí ISP a návrhem dohledového a administrativního systému pro tento typ počítačových sítí. Pro realizaci byla stěžejní volba programovacího jazyka. Po rozboru byl zvolen pro realizaci programovací jazyk Python v kombinaci s PHP pro webovou aplikaci. Možnosti Pythonu jsou velmi rozsáhlé. Python byl navržen tak, aby umožňoval tvorbu rozsáhlých, plnohodnotných aplikací. Kód programu je ve srovnání s jinými jazyky krátký a dobře čitelný. K jeho čitelnosti jednoznačně přispívá nutnost dodržování správného odsazení příkazů dle struktury programu. Vývoj aplikací navíc usnadňuje samotné pojetí jazyka, kdy není nutná jakákoli kompilace zdrojového kódu, neboť spustitelným souborem je samotný zdrojový kód.

Výsledkem práce je plně funkční dohledový systém umožňující správci sítě kontrolu odezvy a ztrátovosti v celé síti. Dále zjišťování a zaznamenávání dat získávaných pomocí protokolu SNMP ze síťových prvků, jako například množství přenesených dat prvkem aj. Vše je možné zobrazit ať už v souhrnných přehledech či v grafech za období až jednoho roku. Navržený systém v základním nastavení zvládne testovat více než 500 síťových prvků a to vše při minimálním vytížení počítače na němž je provozován. Systém byl navrhován pro co největší nezávislost na použitých prvcích sítě, a proto je jediným požadavkem k testování prvku, aby měl přidělenu IP adresu a reagoval na ICMP pakety, či podporoval protokol SNMP pro získávání jiných údajů. Protokol SNMP je možné také nahrazovat získáváním dat pouze TCP spojení na daný prvek, který v tomto případě musí spojení obsloužit zasláním požadovaných dat.

Druhou administrativní částí systému je plnohodnotná klientská databáze. Zde je možné spravovat klienty, jim poskytované služby, platby, přípojky a IP adresy. Klientská databáze je navržena s ohledem na možnosti získávání statistických údajů pro potřeby návrhu celé sítě ISP. Pro účely kontrolování plateb je systémem nabízen import výpisů z banky. Ke klientské databázi je přistupováno pomocí zabezpečené webové aplikace,aby nebylo nutné pořizování jakéhokoli nadbytečného software. Navíc je celá webová aplikace navržena pro využití asynchronní komunikace s webovým serverem, čímž je sníženo množství dat přenášených mezi klientem a serverem. A tímto je aplikace svými odezvami přiblížena běžným počítačovým aplikacím.

Výsledný systém je navíc navržen pro tzv. přídavné moduly, které umožňují rozšíření systému o takřka jakékoli další funkce.

LITERATURA

[1] BOLDIŠ, P. *Bibliografické citace dokumentů podle ČSN ISO 690 a ČSN ISO 690-2* [online]. 2001, poslední aktualizace 11. 11. 2004 [cit. 3. 5. 2008]. Dostupné z URL: <http://www.boldis.cz/citace/citace.html>.

[2] PYTHON, P. *Oficiální stránky Pythonu* [online]. 2007, posledn? aktualizace 11. 11. 2007 [cit. 3. 5. 2008]. Dostupný z URL: <http://www.python.org/>.

[3] OETIKER, T. *Oficiální stránky projektu RRDTool* [online]. 2007, poslední aktualizace 25. 9. 2007 [cit. 3. 5. 2008]. Dostupný z URL: <http://oss.oetiker.ch/rrdtool/index.en.html>.

[4] POSTGRESQL. *Oficiální stránky projektu PostgreSQL* [online]. 2007, poslední aktualizace 25. 9. 2007 [cit. 3. 5. 2008]. Dostupný z URL: <http://www.postgresql.org/>.

[5] ZEDULKA, J. Doc.Ing.CSc. *Konceptuální modelování a návrh databáze* [online]. 2004, poslední aktualizace 3. 12. 2004 [cit. 3. 5. 2008]. Dostupný z URL: <http://www.fit.vutbr.cz/study/courses/DSI/public/pdf/nove/2_kmod.pdf>.

[6] ETINGOF, I. *Modul PySNMP pro python* [online]. 2006, poslední aktualizace 3. 11. 2006 [cit. 3. 5. 2008]. Dostupný z URL: <http://pysnmp.sourceforge.net/>.

[7] ZEND TECHNOLOGIES, Ltd. *Dokumentace PHP* [online]. 2008, poslední aktualizace 16. 5. 2008 [cit. 3. 5. 2008]. Dostupný z URL: <http://www.php.net/manual/en/>.

[8] TRUBETSKOY, G. *Dokumentace projektu mod_python* [online]. 2007, poslední aktualizace 29. 1. 2007 [cit. 3. 5. 2008]. Dostupný z URL: <http://www.modpython.org/live/current/doc-html/>.

[9] PUŽMANOVÁ, R. *Moderní komunikační sítě od A d Z* Computer Press, a.s., 2005, ISBN: 80-251-1278-0

[10] Bigelow, S.J. *Mistrovství v počítačových sítích* ComputerPress a.s, 2004, ISBN: 80-251-0178-9

[11] Mhapsekar, A. *Návody pro tvorbu webových stránek* [online]. 2008, poslední aktualizace 3. 5. 2008 [cit. 3. 5. 2008]. Dostupný z URL: <http://www.w3schools.com/>.

[12] Apache Software Foundation *Dokumentace webového serveru Apache 2.2* [online]. 2008, poslední aktualizace 3. 5. 2008 [cit. 3. 5. 2008]. Dostupný z URL: <http://httpd.apache.org/docs/2.2/>.

SEZNAM SYMBOLŮ, VELIČIN A ZKRATEK

ACID automatizace, konzistence, izolace, trvanlivost – Atomicity, Consistency, Isolation, Durability

AJAX asynchronní javascript a XML – Asynchronous JavaScript and XML

ARP protokol pro rozlišení síťových adres – Address Resolution Protocol

ICMP Internet Control Message Protocol

ISP poskytovatel internetových sluďžeb – Internet Service Provider

HTML značkovací jazyk pro hypertext – HyperText Markup Language

MAC jedinečný identifikátor síťového zařízení – Media Access Control

MIB Management Information Base

MRTG Multi-Router Traffic Graber

NAT překlad IP adres – Network Address Translation

OID identifikátor objektu – Object Identifier

OS operační systém – Operating System

PHP rekurzivní akronym pro „PHP: Hypertext Preprocessor“

ping Packet InterNet Groper

RRA Round Robin Archive

RRD Round Robin Database

SMS krátká textová zpráva šířená pomocí telefonu – Short Message Service

SNMP jednoduchý protokol pro správu – Simple Network Management Protocol

SQL standardizovaný dotazovací jazyk používaný pro práci s daty v relačních databázích – Structured Query Language

SSH Secure Shell

TCP spojově orientovaný přenosový protokol – Transmission Control Protocol

TCP/IP Transmission Control Protocol / Internet Protocol

UML grafický jazyk pro vizualizaci, specifikaci, navrhování a dokumentaci programových systémů – Unified Modeling Language

XML eXtensible Markup Language

SEZNAM PŘÍLOH

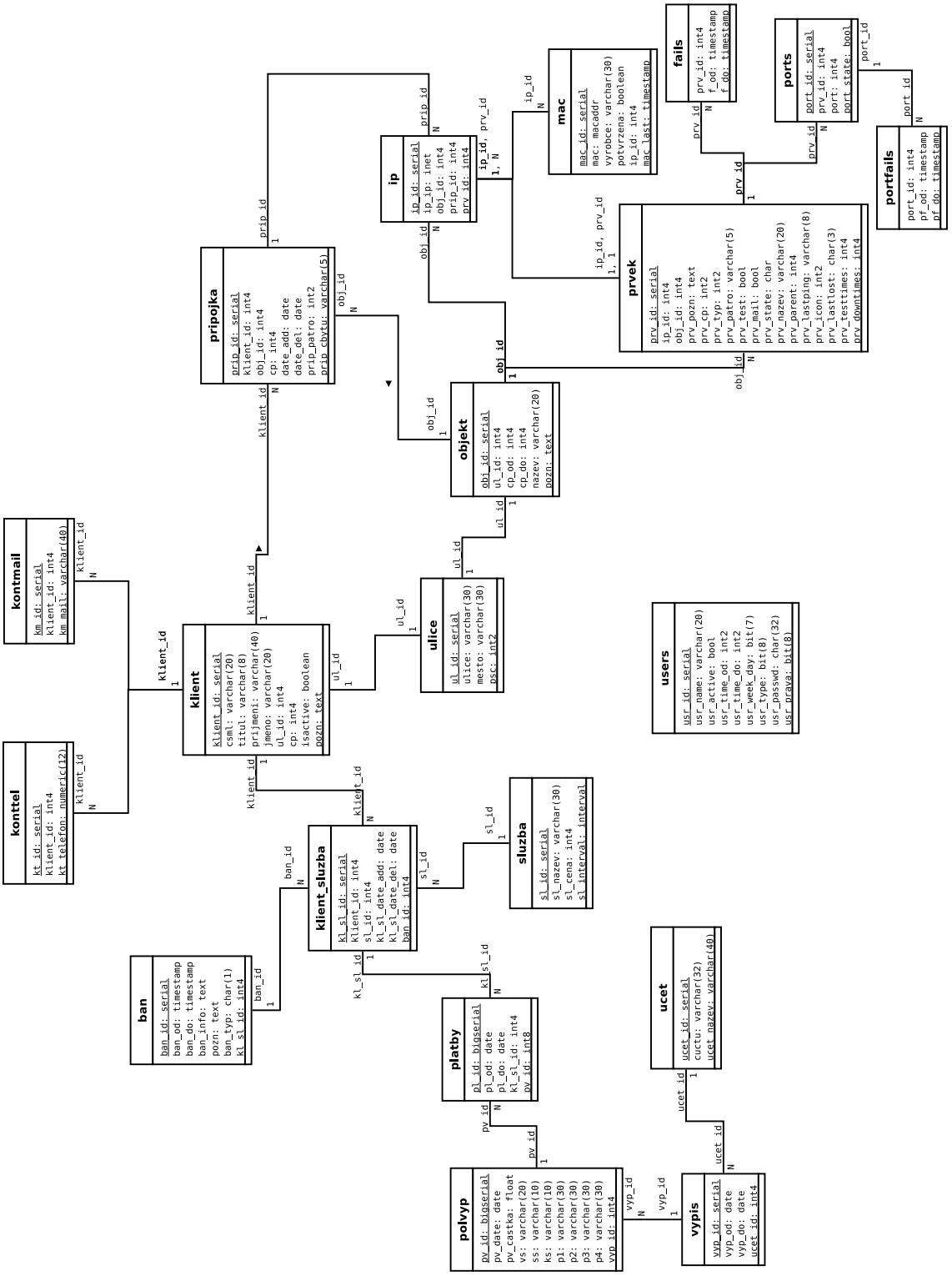
A Schéma navržené SQL databáze **62**

B Webové stránky aplikace **63**

B.1 Dohledová aplikace NetScan 63

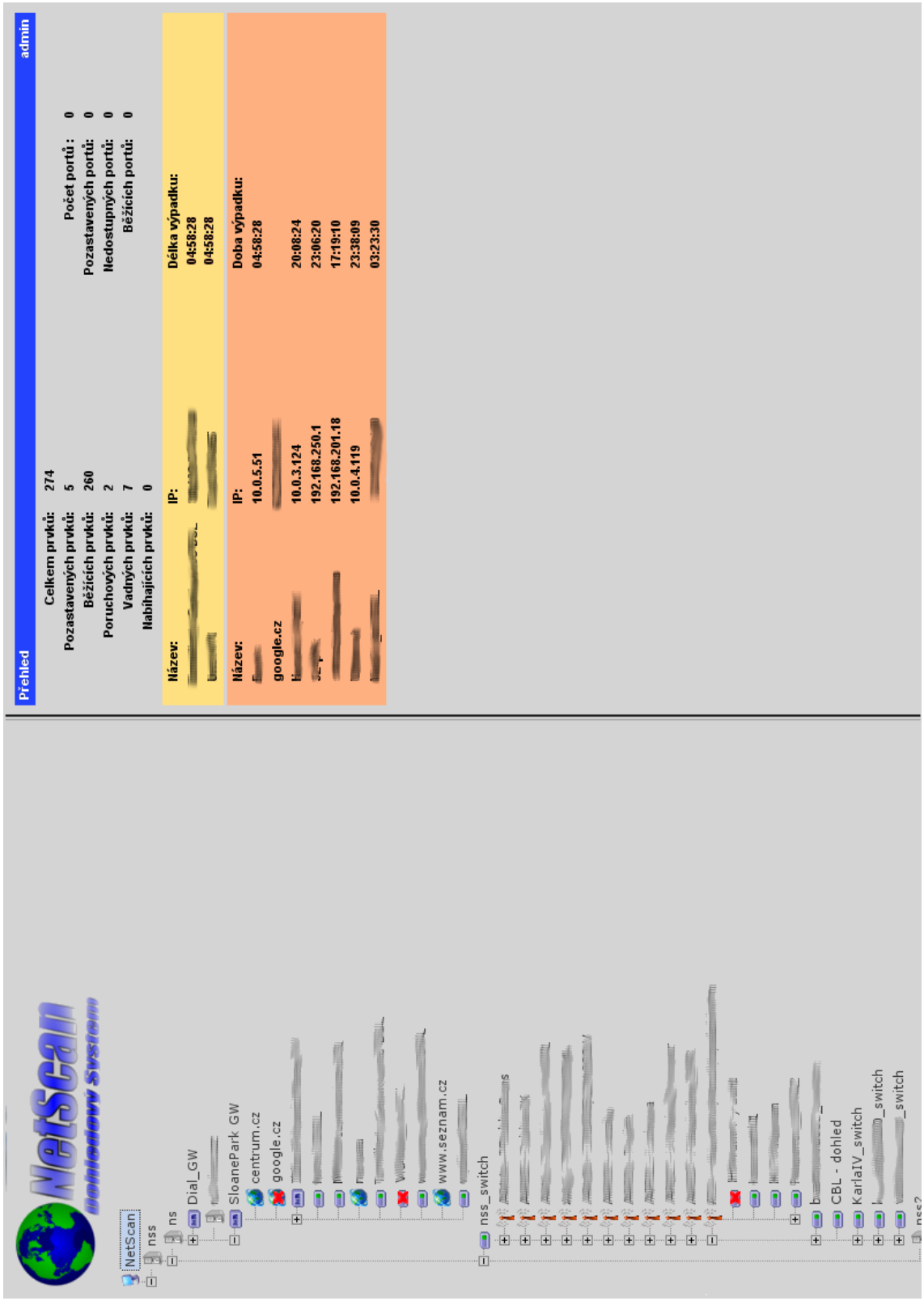
B.2 Přehled síťového prvku 64

A SCHÉMA NAVRŽENÉ SQL DATABÁZE



B WEBOVÉ STRÁNKY APLIKACE

B.1 Dohledová aplikace NetScan



B.2 Přehled síťového prvku