UNIVERZITA PARDUBICE
Fakulta elektrotechniky a informatiky

Sledovací systém dostupnosti směrovačů na síti

Lukáš Kuchta

Bakalářská práce
2010

Univerzita Pardubice
Fakulta elektrotechniky a informatiky
Akademický rok: **2009/2010**

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

(PROJEKTU, UMĚLECKÉHO DÍLA, UMĚLECKÉHO VÝKONU)

Jméno a příjmení:	**Lukáš KUCHTA**
Osobní číslo:	**I07916**
Studijní program:	**B2646 Informační technologie**
Studijní obor:	**Informační technologie**
Název tématu:	**Sledovací systém dostupnosti routerů na síti**
Zadávající katedra:	**Katedra informačních technologií**

Zásady pro vypracování:

Teoretická část:
* Vytvořte přehled technologií pro sledování dostupnosti síťových prvků, zejména routerů.
* Porovnejte technologie z hlediska náročnosti na SW a HW (centrálního prvku i routerů samotných) a z hlediska možností a vlastností.
* Zaměřte se na produkty, které jsou zcela nenáročné na SW na straně sledovaného prvku a vyberte z nich vhodné kandidáty.

Praktická část:
* Implementujte řešení pro sledování dostupnosti síťových prvků (routerů) se zaměřením podle bodu 1.3 zadání.
* Navrhněte a vytvořte databázi pro ukládání stavu dostupnosti v určitých časových intervalech.
* Navrhněte a vytvořte přehlednou vizuální interpretaci zjištěného stavu s možností náhledu na stav v historii.

Pozn.: Uvažujte, že sledované síťové prvky jsou:
1. dané seznamem svých adres,
2. dané rozsahem síťových adres. V tomto případě je třeba automaticky zjistit (i průběžně periodicky či na vyžádání) seznam existujících prvků.

V obou případech uvažujte i seznam adres dočasně či trvale vyřazených z aktivního sledování.

Rozsah grafických prací:

Rozsah pracovní zprávy:

Forma zpracování bakalářské práce: **tištěná/elektronická**

Seznam odborné literatury:

* **Kolektiv autorů: LINUX dokumentační projekt, Computer Press 1998 (3. aktualizované vydání), ISBN 80-7226-761-2**
* **Shah, S.: Administrace systému Linux ? podrobný průvodce začínajícího administrátora, Grada Praha 2002, ISBN 80-7169-586-6**
* **manuálové stránky unixových příkazů ping, traceroute, netcat**
* **MikroTik: Dude: http://wiki.mikrotik.com/wiki/MikroTik_Dude**

Vedoucí bakalářské práce: **Mgr. Tomáš Hudec**
Katedra informačních technologií

Datum zadání bakalářské práce: **15. ledna 2010**

Termín odevzdání bakalářské práce: **14. května 2010**

L.S.

prof. Ing. Simeon Karamazov, Dr.
děkan

Ing. Lukáš Ogan, Ph.D.
vedoucí katedry

V Pardubicích dne 31. března 2010

Prohlášení

Prohlašuji:

Tuto práci jsem vypracoval samostatně. Veškeré literární prameny a informace, které jsem v práci využil, jsou uvedeny v seznamu použité literatury.

Byl jsem seznámen s tím, že se na moji práci vztahují práva a povinnosti vyplývající ze zákona č. 121/2000 Sb., autorský zákon, zejména se skutečností, že Univerzita Pardubice má právo na uzavření licenční smlouvy o užití této práce jako školního díla podle § 60 odst. 1 autorského zákona, a s tím, že pokud dojde k užití této práce mnou nebo bude poskytnuta licence o užití jinému subjektu, je Univerzita Pardubice oprávněna ode mne požadovat přiměřený příspěvek na úhradu nákladů, které na vytvoření díla vynaložila, a to podle okolností až do jejich skutečné výše.

Souhlasím s prezenčním zpřístupněním své práce v Univerzitní knihovně.

V Pardubicích dne 28. 04. 2010

Lukáš Kuchta

Poděkování

Děkuji tímto panu Mgr. Tomáši Hudcovi za odborné vedení a pomoc při tvorbě této bakalářské práce a hlavně za trpělivost a čas, který mi věnoval.

Abstrakt

Cílem teoretické části této bakalářské práce je vytvořit přehled technologií pro monitoring síťových zařízení. Porovnat dostupné technologie z hlediska náročnosti na sledovaný i sledující prvek s ohledem na možnosti a vlastnosti, které daná technologie nabízí. V závěru teoretické části je výběr produktů, které jsou co nejméně softwarově náročné.

Praktická část pak zahrnuje implementaci monitorovacího systému s důrazem na co nejmenší náročnost sledovaného prvku.

Klíčová slova

monitoring, router, dostupnost, SNMP, ICMP, RMON, ping

Title

Watchdog system for monitoring reachability of routers

Abstract

The theoretical part of this thesis is a summary of technologies for monitoring network devices. Compareation of the available technologies in terms of performance with respect to options and properties that the technology offers. In conclusion, the theoretical part is to choose products that are the least demanding software.

The practical part includes the implementation of a monitoring system with the emphasis on the minimum load of the reference element.

Keywords

monitoring, router, reachability, SNMP, ICMP, RMON, ping

Obsah

Úvod

V rozsáhlých počítačových sítích je dost velký problém odhalit výpadek daného zařízení a příčinu, která výpadku předcházela, jako například ztráty na lince, nebo vzrůstající doba odezvy. Tyto události je potřeba včas odhalit a právě to je cílem této bakalářské práce, vytvořit systém, který daný problém řeší. Jako jeden z cílů, které jsem si stanovil, byla platformová nezávislost. Dále pak snadná možnost doplnění na komplexní monitorovací systém. A také minimalizovat zátěž při sledování, aby nedocházelo ke zbytečnému vytěžovaní sledujícího systému zbytečnými úkony. Při implementaci bude mým cílem snaha o intuitivní ovládání a přívětivý přístup k uživateli, který bude aplikaci využívat.

1. Monitoring jako pomocník administrátora

Pod pojmem monitoring si lze představit dohled či sledování určitého systému, který je pro nás svým provozem a funkčností důležitý. Monitorování je tedy sledování nějakého stavu. V síťovém prostředí je, při dohledu nad infrastrukturou, klíčové sledovat zejména dostupnost směrovačů, jejichž důležitost byla zmiňována v úvodu.

1.1. Účastníci monitoringu

V monitorovacím procesu se nachází vždy více zařízení, pokud nebereme v potaz, že zařízení sleduje samo sebe. Definujme si tedy pojem **host** ve smyslu sledovaného prvku a pojem **server** ve smyslu dohlížejícího prvku. Na serveru bývá obvykle databáze v podobě uložených dat o sledovaných hostech, kterou je možno využít ke tvorbě statistik a podrobit různým analýzám.

1.2. Na co zaměřit monitoring

Rozsah sledovaných proměnných je celá řada. V mé bakalářské práci se zaměřuji výhradně na dostupnost směrovačů, ale pro správné pochopení situace, je potřeba si přiblížit i další možnosti.

Souhrn možností pro oblast monitoringu

- Dostupnost zařízení,
- dostupnost služeb,
- průtok dat,
- události (logy) vyskytující se na sledovaném prvku,
- vytížení systémových zdrojů,
- zvláštní veličiny, jako jsou informace z čidel průmyslových zařízení.

Znalost těchto veličin nám napomáhá v rozhodování a plánování budoucího rozvoje síťové infrastruktury.

1.3. Tvorba monitorovacího systému

Lze využit dvě možnosti, jak uskutečnit požadovaný cíl. Jedna z nich je ta, že použijme stávající **hotová řešení**, komerční nebo volně dostupná. Druhá pak je **návrh a tvorba vlastního** řešení, na což se zaměřím v praktické části této práce. Obě řešení mají bezesporu řadu výhod i nevýhod, které plynou z jejich podstaty.

1.3.1. Možnosti implementace

Dohled pomocí vlastního démona, který je realizován tak, že na sledovaný cíl je umístěn vlastní kód, který zajišťuje sběr a distribuci požadovaných informací. Toto je vhodné zejména v případě, kdy si nevystačíme s dostupnými prostředky, například sběr dat z programovatelných automatů PLC [2].

Nedostatky

- Nutný zásah na sledovaný prvek,
- platformově závislé,
- možná zátěž sledovaného prvku,
- možnost nekompatibility s jiným softwarem.

Výhody

- Sběr veškerých potřebných informací.

Dohled bez pomoci vlastního démona lze realizovat pomocí standardních protokolů, jež dané zařízení podporuje. Vhodné ve většině případů u zařízení v kategorii IP sítí. Toto řešení podporuje většina výrobců síťového hardwaru [2].

Nedostatky

- Ne vždy může poskytnout dostatečné informace,
- specializovaná zařízení nemusí tuto možnost podporovat.

Výhody

- Není nutný žádný zásah, nebo jen minimální, na sledovaný prvek,
- minimalizace zátěže,
- platformově nezávislé, jde většinou o standard.

2. Technologie ke zjištění dostupnosti směrovačů

Ve větších sítích je přímo nutností mít přehledný stav dostupnosti jednotlivých zařízení, na nichž je infrastruktura sítě závislá. Při větším počtu prvků bez monitoringu může někdy trvat velice dlouho zjištění, kdy a kde konkrétně nastal daný problém, a to vzhledem k navazujícím spojům nebo různým cestám mezi směrovači. Rozhodl jsem se zaměřit na protokol IP, protože je nejužívanější. Dále se budu věnovat technologiím nad tímto protokolem použitelné.

2.1. Protokol ICMP (Internet Control Message Protocol)

2.1.1. Stručný charakter

Jedná se o protokol, který je součástí přímo protokolu IP. Každá implementace IP musí podporovat i tento servisní protokol. Důvodem jeho existence je informovat o chybách při přenosu sítí. Pomocí ICMP lze ověřit status rozhraní zařízení a zjistit tak jeho dostupnost. Zjištění stavu probíhá pomocí vyslání zprávy ICMP **echo request** a očekává se odpověď ICMP **echo reply**, přičemž lze změřit dobu prodlevy mezi vysláním žádosti a příjmem odpovědi. Hojně využívané nástroje stavěné na tomto protokolu jsou nástroje ping a traceroute, které jsou běžně dostupné v určitých modifikacích snad na všech operačních systémech [1].

2.1.2. Popis nejužívanějších typů ICMP

- **Echo Request** – jedná se o požadavek na odpověď. Každý prvek implementující IP protokol musí být schopen na tuto žádost odpověď. Ne vždy se tomu tak děje, a je to z nepochopitelných důvodů v určitých programech, či přímo operačních systémech zakázáno. Bylo by to pochopitelné u některých koncových systémů obsahujících citlivá nebo utajovaná data, ale ve většině případů je to dle mého úsudku, špatně.
- **Echo reply** – je odpověď na výše zmíněný požadavek, na který je cílový host povinen odpovědět.
- **Destination Unreachable** – je typ datového paketu (datagramu), z kterého lze vyčíst další informace, například o nedostupnosti cílové sítě nebo hosta.
- **Time Exceeded** – informace o tom, že během přenosu došlo ke snížení hodnoty TTL (Time To Live) na 0.

Cílem této práce není detailní popis protokolů, proto jsem uvedl jen některé z vlastností a typů protokolu ICMP, na základě nichž lze pak rozhodnout o vhodnosti výběru při implementaci.

2.1.3. Nedostatky

- Lze pomocí něj ověřit jen omezené množství informací, de facto jen ztrátovost a dobu odezvy,
- možnost úmyslné blokace, což znemožňuje jeho použití,
- sám o sobě neumožňuje žádnou formu zabezpečení.

2.1.4. Výhody

- Přítomnost na každém prvku podporující protokol IP,
- podpora přímo v OS, a tím pádem není nutný žádný obslužný démon,
- minimální hardwarová i softwarová náročnost,
- není nutná žádná konfigurace.

2.1.5. Programy využívající protokol ICMP

Ping

Program sloužící k ověření, zda na daném zařízení je funkční protokol IP.

Traceroute

Program slouží ke zjištění směrovačů na cestě k požadovanému cíli. Výstupem je pak jejich seznam. Traceroute posílá pakety UDP (protokol je možno změnit vhodným přepínačem) a postupně inkrementuje hodnotu TTL o jedna (začíná se od jedničky) a očekává chybovou zprávu ICMP o vypršení limitu, kterou vygeneruje směrovač, jenž hodnotu TTL musel zmenšit o jedna na nulu. Postupně si takto sestavuje seznam uzlů na cestě k cíli [4].

2.2. Protokol SNMP (Simple Network Management Protocol)

2.2.1. Stručný charakter

Jedná se o protokol založený na architektuře klient/server. V terminologii SNMP je serverem chápan termín **agent**, a pod pojmem klient je myšlen výraz **manager.** V typické situaci model SNMP funguje tak, že manager posílá požadavky na agenta a ten mu na ně patřičně odpovídá. Je zde ale ještě jiná vlastnost, která se od klasického modelu kli-

ent/server liší, a to je ta, že agent může sám vyslat zprávy na předem definované adresy IP a to i v případě, vyskytne-li se nějaký podmíněný stav. V takovém případě agent generuje tzv. **trap**, čemuž lze rozumět jako vyvolání výjimky, kterou by měl zachytit a zpracovat manager. Je zde ale problém, že agent nedostane informaci o tom, že manager vyvolaný trap obdržel či nikoliv. Tedy, neplatí tvrzení, že když manager nedostává trap, je vše v pořádku [9].

Pomocí SNMP lze získávat pouze okamžité hodnoty. Pro možnost vytvářet grafické vizualizace a udržovat historii stavů je tedy nutné agenty periodicky dotazovat a ukládat získané stavy na straně managera, nejlépe do databáze, kterou lze pak následně podrobit analýze a zpracovat [11].

SNMP je postaven nad protokolem UDP, díky čemuž dosahuje vyšších rychlostí oproti spojovaným protokolům. Zařízení ani nemusí obsahovat protokol TCP, který obsahuje poměrně složitý stavový mechanismus. Ovšem rychlost lze zaručit na úkor kvality a nelze tedy řešit kontrolu doručení dat vzhledem k nespojovému charakteru protokolu UDP, ale na druhou stranu díky tomuto řešení, lze využít i méně výkonné mikroprocesory [10], [11].

2.2.2. Agenti

Agenti neposkytují žádné grafické rozhraní a slouží jen ke sběru a přenosu informací. Měl by být malý a jednoduchý z důvodu minimálního vlivu na funkci monitorovaného zařízení, na kterém se nachází. Přijde-li tedy agentovi dotaz, zareaguje na něj tak, že vyhledá patřičnou informaci v **databázi MIB** a odpoví managerovi [12], [13].

2.2.3. Databáze MIB

Základem protokolu SNMP jsou proměnné uspořádané do hierarchického stromu (datová struktura). Tyto proměnné odrážejí stav prvků daného zařízení. Každá hodnota je jednoznačně identifikována pomocí číselného identifikátoru nazývaného OID (Object Identifier). Identifikátor je tvořen posloupností čísel oddělených tečkovou notací. Čísla vyjadřují úrovně ve stromu. MIB (Management Information Base) je kolekce informací složená z objektů. Databáze obsahuje jména a popisy jednotlivých hodnot OID [10].

2.2.4. Verze

V současné době je protokol SNMP dostupný ve třech verzích.

SNMP verze 1

- Obsahuje pouze 5 operací
    - **GetRequest** – žádost managera o informace z agenta,
    - **GetNextRequest** – žádost o další informace,
    - **GetResponse** – agent vysílá odpověď na žádost,
    - **SetRequest** – nastavuje hodnotu proměnné v MIB agenta,
    - **Trap** – agent vysílá oznámení nějaké události.
- Ochrana pomocí textového řetězce.
- Odlišný formát zprávy mezi Trap a GetRequest, GetNext, SetRequest, GetResponse. Konkrétní odlišnosti v části PDU (Protocol Data Unit).

SNMP v2

- Přidána funkce inform, sloužící ke komunikaci mezi managery,
- možnost využít dotaz (GetBulk), který vrátí více záznamů, čímž ušetříme náklady na přenos dat,
- sjednocení formátu zpráv.
- Tato verze se příliš neujala, z důvodů neshod ohledně složitosti implementace. Nejrozšířenější verze dvojkové řady je označována jako verze 2c.

SNMP v3

- Možnost šifrování zpráv,
- zvýšení bezpečnosti, možnost autentizace a integrity zpráv jméno/heslo, otisk MD5/SHA1,
- změna formátu zprávy s ohledem na bezpečnost,
- možnost omezit přístupy k MIB objektům.

2.2.5. Nedostatky

- Nutný obslužný program (agent),
- u některých verzí primitivní způsob ochrany,
- díky obslužnému programu vyšší nároky na systémové prostředky,
- nutnost konfigurace,
- v případě výpadku linky přijdeme o možnost měřit hodnoty.

2.2.6. Výhody

- Široké spektrum získávaných informací,

- jedná se o de facto standard, takže je podporován řadou výrobců.

2.2.7. Programy využívající protokol SNMP

Zaměřím se na stručný popis linuxových nástrojů obsažených v balíčku Net-SNMP, protože je na řadě linuxových distribucí a jedná se prakticky o nejrozšířenější nástroje. Příkazy typu **get** lze chápat jako možnost čtení a analogický příkaz **set** jako možnost zápisu.

Obsah balíčku:

- **snmpget** – nástroj pro posílání SNMP dotazů na agenta,
- **snmpwalk** – umožňuje průchod podstromem MIB,
- **snmpbulkget** – dotaz, který vrací více údajů najednou,
- **snmptrap** – generuje signál z agenta a očekává se jeho zpracování managerem,
- **snmptrapd** – démon zachytávající trapy,
- **snmpd** – démon zajišťující funkci agenta,
- **snmpset** – dotaz k zapsání hodnoty do MIB agenta,
- **snmptranslate** – umožňuje převod mezi číselným a textovým popisem usnadňující přehlednost.

2.3. RMON MIB (Remote Network MONitoring)

2.3.1. Stručný charakter

Vzhledem k tomu, že SNMP umožňuje čtení pouze aktuálních hodnot, což při výpadku linky vede k nedostupnosti požadovaných informací, bylo nutné tento nedostatek odstranit a o to se postaral právě standard RMON. Stejně jako u SNMP jde o model klient/server, kde server je v tomto kontextu označován jako **sonda**. Sondy mohou být integrovány přímo do síťového zařízení nebo se mohou vyskytovat jako samostatné prvky. Vzhledem k tomu, že RMON je převážně zaměřen na sběr informací o toku dat, zaměřím se na něj jen okrajově.

RMON nabízí možnost odklonit převážnou část výpočtů z managera na sondu. Sonda totiž umožňuje uchovávat i historické údaje a statistiky. Toto řešení umožňuje jednak ulehčit přenosové lince, a to tak, že manager si vyžádá komplexní data na rozdíl od SNMP, kdy je nutné se ke stejným výsledkům dostat periodickým dotazováním agenta, a

jednak ulehčíme i výpočetní výkon managera. Sondy je možné dotazovat pomocí SNMP managera a dostávat tak z nich potřebné informace [14].

V současnosti existují dvě verze. První verze umožňuje práci pouze na linkové vrstvě, což je ve směrovaných sítích nepoužitelné, proto vznikla verze 2, která rozšiřuje působnost až na aplikační vrstvu.

2.3.2. RMON MIB

Standard specifikuje MIB jako datové struktury SNMP obsahující statistické tabulky. Tabulky jsou zde chápány jako skupiny (**groups**). U RMON 1 a sítí typu ethernet rozeznáváme devět skupin [15].

Skupiny pro Ethernet:

- **ethernet statistic** – udržuje statistiky o provozu a chybách v segmentu,
- **ethernet history** – historický pohled na statistiku poskytovanou první skupinou,
- **alarms** – nastavení prahových hodnot pro generování výjimek při jejich překročení,
- **hosts** – statistiky vztažené ke stanicím,
- **host top n** – informace o vybraných stanicích podle určitého kritéria,
- **traffic matrix** – údaje o komunikaci mezi dvojicí zařízení,
- **filters** – mechanismus pro výběr paketu dle určitých podmínek,
- **packet capture** – zachytávání paketu dle pravidel z filtru,
- **events** – řídí vysílání poplašných zpráv a provádí akce.

2.3.3. Nedostatky

- Nutnost obslužného programu (sondy),
- sondy jsou náročnější oproti SNMP agentům,
- sledování velkého množství údajů může být náročné na systémové prostředky,
- ne všechna zařízení tento standard podporují.

2.3.4. Výhody

- Snížení nároků na přenosovou linku,
- odlehčení managerovi.

2.4. Vlastní řešení

Jedná se o metodu, po které sáhneme v případě, kdy nám hotová řešení nevyhovují. Lze to realizovat tak, že si navrhneme **vlastní protokol**, například nad protokoly TCP, UDP nebo využijeme **dostupné diagnostické nástroje**, a sestavíme z nich komplexní řešení splňující naše potřeby. **První možnost** vyžaduje vysoké odborné znalosti dané tématiky a také čas pro potřebnou realizaci. Touto metodou se zde nebudu zabývat vzhledem k náročnosti a povaze problému. Nároky na HW a SW zde také hodnotit nebudu, protože ty se různí implementace od implementace. **Druhá možnost** nám pak nabízí poměrně pružné rozhodování ve vývoji, kdy můžeme poskládat pomocí programovacích jazyků již vytvořené diagnostické nástroje do celku, který nám zajistí potřebné služby. Hodnotit zde budu pouze druhou možnost.

2.4.1. Nedostatky

- Znalosti programování a potřebných nástrojů (relevantní),
- čas potřebný k realizaci závislý na rozsahu řešení.

2.4.2. Výhody

- Prakticky téměř neomezené možnosti.

3. Výběr vhodné technologie dle zadání

Zhodnotím-li výše zmíněné technologie a vyberu-li z nich nejvhodnějšího kandidáta podle kriterií ze zadání, vychází mi nejlépe použití protokolu **ICMP v kombinaci s metodou vlastní implementace** s použitím dostupných diagnostických nástrojů. Nároky jednotlivých technologií jsou zmíněny výše v textu, kdy jsem se jednotlivým technologiím věnoval podrobněji a na základě nich jsem se rozhodl použít právě tento protokol.

4. Vlastní implementace

Pracovní název projektu jsem zvolil **Monitor**, tak se na něj v průběhu práce budu odkazovat. Při implementaci jsem se snažil použít co nejméně potřebných programů, aby bylo možné Monitor snadno a bez větších úprav portovat i na jiné systémy. Tato práce byla vytvořena konkrétně pro systém GNU/Linux. Průběh implementace probíhal tak, že jsem nejprve vytvořil monitorovací skripty a následně pak webové rozhraní pro snazší ovládání a grafické znázornění naměřených výsledků.

Celá monitorovací část je psána pomocí skriptů pro interpret BASH v kombinaci s programovacím jazykem AWK. Volil jsem programovou výbavu jednak dle zkušeností, ale také dle toho, že jsou oba zvolené komponenty součástí prakticky snad všech distribucí, a není nutné nic dodatečně instalovat. Webová část je pak vytvořena v PHP/HTML.

4.1. Sběr dat

Jako první, na co jsem se zaměřil, byl sběr potřebných dat a k tomu jsem se rozhodl použít linuxového programu **PING**, který mi potřebné informace poskytl. Prvotní myšlenka byla sledovat pouze dostupnost, tedy dva stavy. Což ovšem nemá velkou vypovídající hodnotu, protože nezjistíme, co případné nedostupnosti předcházelo. Může to být například vzrůstající doba odezvy nebo ztráty na lince. Program PING, je obsažen např. v balíčku iputils, jenž je součástí prakticky všech dnes vydávaných distribucí. Rozhodl jsem se pro sběr **čtyř veličin,** dle kterých lze vznikající problém odhalit.

4.1.1. Měřené veličiny

- Minimální doba odezvy,
- maximální doba odezvy,
- průměrnou doba odezvy,
- ztráta dat na lince v procentech.

Přičemž poslední uvedená veličina není v grafech zakreslena, je to spíše pro budoucí možnost rozšíření. Takže tedy ztráta je ukládána, ale nedochází k její grafické prezentaci. O sběr dat se stará následující kód.

```
ping -n IP_ADRESA -i0.2 -c3 -w1 -q | tail -n2 | awk '
{
  if(NR == 1){
    gsub("%","",$6);
```

```
    ztrata = $6;
  }else{
    delkaPole = split($4,odezva,"[/]");
  }
}

END{
  if(delkaPole!=0){
    print odezva[1] ":" odezva[2] ":" odezva[3] ":" ztrata;
  }
}'`;
```

Parametry programu ping:

- **-n** – program nepřekládá adresy IP na jejich doménová jména. To může trvat určitou dobu, navíc v programu se jmény nepracuji, tak jsem jejich překlad vypustil kvůli vyšší rychlosti zpracování.
- **-i** – jedná se o interval mezi jednotlivým posíláním paketů. Menší hodnoty než 0,2 lze použít pouze s právy super uživatele. Volil jsem tedy hodnotu 0,2; aby bylo možné používat příkaz i bez super uživatelských práv.
- **-c** – parametr určující počet za sebou vyslaných paketů, po kterých program samovolně skončí. Hodnotu 3 jsem zvolil tak, aby byla doba testu aktivního zařízení kolem 0,5 s.
- **-w** – doba v sekundách, po kterou ping ukončí svou činnost. U dostupných zařízení se dosáhne výsledu zhruba za 0,5 s, takže je 1s interval dostačující. U nedostupných prvků pak končí běh testu právě po uplynutí jedné sekundy.

Kombinace parametru **-c** a **-w** zajistí, že dostupné zařízení je otestováno za **0,5 s** a test je ukončen. U nedostupného zařízení končí test za **1 s**. V případě, že je parametr **-w** vynechán, trvá test nedostupného zařízení **10 s,** což je při vyšším počtu testovaných zařízení dlouhá doba.

AWK program se pak postará o výstup čtyř hodnot oddělených dvojtečkou. Jedná se o hodnoty:

- minimální odezva,
- průměrná odezva,
- maximální odezva

- ztrátovost.

4.2. Ukládání dat

V aplikaci jsou uvažovány data dvojího charakteru a to jednak **data zadaná uživatelem**, v tomto smyslu je uživatel chápan jako správce, který zadá jednotlivé adresy IP nebo rozsahy do systému, a jednak pak **data měřená** v průběhu daného období, o jejichž sběr se stará automaticky skript. Měřená data jsou časového charakteru, jde o dobu odezvy k sledovanému zařízení.

Vzhledem k tomu, že jsem na projektu podobného charakteru pracoval poprvé, původní myšlenka byla ta, že budu veškerá data ukládat do databáze SQL. Ovšem to se ukázalo v průběhu vývoje jako ne zrovna šťastné řešení, protože by docházelo k velkému nárůstu datových souborů databáze SQL.

Rozhodl jsem se tedy data rozdělit do dvou výše zmíněných skupin. Pro uživatelská data jsem použil databázi SQL a pro měřená data databázi RRD, která šetří paměťové prostředky.

4.2.1. Uživatelská data

Pro uživatelská data jsem volil databázi SQL a to konkrétně databázový server MySQL. Důvodem byla snadná manipulace, ukládání dat a široké spektrum užitečných funkcí. Adresy IP ukládám v podobě **15 bytového** řetězce, protože datovým typem pro adresy IP tato databáze nedisponuje.

Zjistil jsem, že jiné databázové servery jako například PostgreSQL nabízí datový typ **inet**, který je určen k uchovávání adres IP nebo sítí, což by bylo nejspíš vhodnější, protože má pouze **12 bytovou** délku. Vzhledem k tomu, že s databází MySQL mám vetší zkušenosti, volil jsem právě tu.

Databázový server PostgreSQL nabízí mimo jiné i datové typy jako:

- **cidr** – pro uchovávání sítí IPv4, velikost datového typu je 12B,
- **inet** – pro uchovávání sítí IPv4 a uzlů, velikost datového typu je 12B,
- **macaddr** – pro uchovávání adres MAC, velikost datového typu je 6B.

Problém, na který jsem narazil, bylo řazení adres IP ve správném pořadí při vypisování. Protože řazení řetězců probíhalo tak, že se adresy vypisovaly v pořadí 1, 10, 11, 2, …, ale požadovaný výstup měl být 1, 2, 10, 11, …, rozhodl jsem se využít funkci

INET_ATON, která je součástí přímo jazyka serveru MySQL. Funkce přebírá jako svůj parametr řetězec ve formátu adresy IP a vrací jeho reprezentaci jako celé číslo. Výstup této funkce používám v klauzuli ORDER BY pro řazení [17].

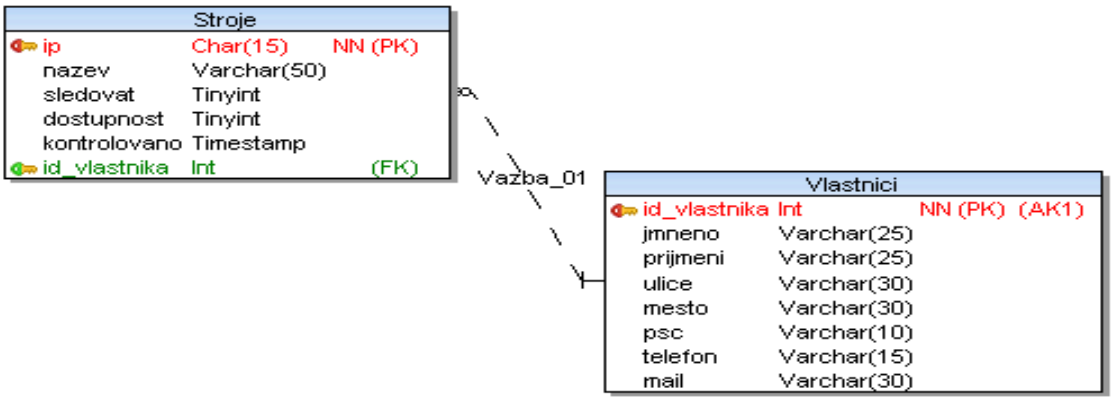
```
SELECT INET_ATON('209.207.224.40');
3520061480
```

Výsledku se pak dosáhne následujícím výpočtem:

$$209 \cdot 256^3 + 207 \cdot 256^2 + 224 \cdot 256^1 + 40 \cdot 256^0$$

Řazení celých čísel pak probíhá přesně tak, jak se očekává.

Databáze SQL se skládá ze dvou tabulek, přičemž tabulka **Vlastníci** je zde spíše jako demonstrativní ukázka toho, že je možné Monitor napojit na některé z existujících systémů obsahujících databázi uživatelů připojených do internetové sítě. Proto jsem nekladl na návrh této tabulky příliš velký důraz, co se týče volby datových typů nebo integritních omezení. V tabulce **Stroje** jsou pak uložena data sledovaných prvků. Datové typy a integritní omezení jsou vidět na **Obrázek 4.2.1**.



Obrázek 4.2.1 – ER diagram SQL databáze Monitoru

4.2.2. Plnění uživatelských dat

Jak bylo zmíněno výše, jedná se o data, která zadal uživatel ručně do databáze, tedy jednotlivé adresy IP sledovaných zařízení prostřednictvím webového formuláře. Z databáze si je pak monitorovací skript přebere a zpracuje.

Vzhledem k tomu, že aplikace měla umožňovat naplnit databázi i aktivními adresami z patřičného rozsahu, bylo nutné nějakým způsobem zajistit automatické plnění, protože rozsah může být poměrně obsáhlý, čímž by ruční plnění bylo značně nepříjemné.

Pro vyřešení tohoto problému jsem se rozhodl použít programy **fping** a **nmap**, které umožňují otestovat celý zadaný rozsah a vypsat zařízení, která jsou aktivní. Nástroje bylo nutné doinstalovat, protože nejsou součástí standardní instalace distribuce.

Při testování se ukázalo, že nmap vychází téměř **5x** rychleji než fping. Měření jsem provedl pomocí nástroje **time** a jeho výstupní hodnoty **real**. Rozsahy jsem zvolil náhodně. Výsledky jsem pak zaznamenal do **Tabulka 4.2.2**. Testovací příkazy vypadají následovně:

```
time nmap -nsP -T5 SÍŤ/MASKA
time fping -g SÍŤ/MASKA
```

Parametr **-T5** u programu nmap slouží k vyladění hodnoty ping timeout. Dle manuálu – čím vyšší číslo, tím rychlejší, což dokazuje i test, který jsem provedl (**Tabulka 4.2.1)** a na základě čehož jsem volil právě -T5**.**

Tabulka 4.2.1 – Test parametru -T programu nmap

Maska sítě	**Parametr**	**Čas**	**Dostupných**
/24	-T0	17min 40,287 s	2
/24	-T1	12min 54,650 s	2
/24	-T2	12min 53,659 s	2
/24	-T3	0min 3,191 s	2
/24	-T4	0min 3,139 s	2
/24	-T5	0min 2,343 s	2

Tabulka 4.2.2 – Porovnání výkonnosti programů nmap a fping

Maska sítě	**Fping**	**Nmap**	**Dostupných**
/24	36,409 s	1,751 s	42
/24	31,243 s	1,699 s	68
/24	40,085 s	2,045 s	23
/24	37,274 s	7,051 s	1
/24	37,291 s	27,053 s	0
/24	39,867 s	1,997 s	18
/24	40,721 s	2,224 s	4
/27	8,406 s	1,240 s	4
Průměr	**33,8 s**	**5,61 s**	

Jako vhodné řešení jsem na základě testu vybral nástroj nmap, který je nutné doinstalovat. Nmap poskytuje oproti fping nesporné výhody. Funkce, která se stará o procházení rozsahu, detekuje aktivní zařízení a zařadí je do databáze SQL. Jako návratovou hodnotu vrací počet nalezených aktivních hostů. Návratovou hodnotu přebírá webová aplikace, která tak informuje uživatele o počtu zařazených prvků do sledování. Po zařazení do databáze SQL, je nad daty proveden test a naměřené veličiny jsou uloženy do databáze RRD.

```
function projdiRozsah(){
  if [ ! -e $TMPF ]; then
    touch $TMPF;
    chmod o+wrx $TMPF;
  fi;
```

```
  $NMAP -T5 -nsP $1/$2 -oG $TMPF &>/dev/null
  I=0;

  for ip in `cat $TMPF | grep Host | awk '{print $2}'`; do
    $MYSQL -e "insert INTO Stroje values
('$ip','Nedefinovan',1,1,NULL,1)" 2>/dev/null;
    I=`expr $I + 1`;
  done;

  echo $I;
}
```

4.2.3. Měřená data

O měřená data se nestará uživatel, ale skript, který je automaticky sbírá. Pro tato data jsem se rozhodl zvolit Round-Robin způsob uchovávání dat, což nabízí oproti SQL výhodu v menší velikosti datových souboru.

4.2.4. RRDtool

RRDtool je flexibilní systém sloužící k ukládání, zpracování a zobrazování číselných údajů v čase. Používá princip RRD (Round-Robin Database), tzn. datový soubor má konstantní velikost a starší data jsou dostupná s nižší rozlišovací schopností. Postupně dochází k přepisování starších dat za nová.[18].

Balíček RRDtool obsahuje několik nástrojů pro tvorbu a aktualizaci datových souborů, a nástroje pro kreslení grafů ze získaných dat. Jako jednu z nevýhod lze považovat to, že nemůžeme přistupovat k datům z konkrétního časového úseku v minulosti (například nemohu získat konkrétní údaje z minulého roku, z daného měsíce a v čase 10:00–11:00), protože RRD tato data agreguje pomocí statistických funkcí. Ovšem pro mé potřeby je použití tohoto nástroje více než dostačující, a uchovávání dat je zde spíše druhořadá záležitost, nástroj je určen spíše ke kreslení grafů ze současných měření.

Databáze RRD se skládá z několika archivů, které mají určitý počet řádků a podobají se tabulkám v SQL. Databáze obvykle obsahuje více archivů s tím, že se jednotlivé archívy liší ve velikosti ukládaných časových úseků. Archivy se vytváří při vytvoření databáze. To, který archiv se použije, vybere RRDtool automaticky s ohledem na kreslené období a přesnost.

Obecně se databáze RRD a jednotlivé archívy RRA (Round Robin Archive) vytváří následujícím příkazem.

```
rrdtool create filename [--start|-b start time] [--step|-s step]
 [DS:ds-name DST:heartbeat:min:max] [RRA: CF:xff:steps:rows]
```

- **filename** – je název souboru, ve kterém je celá databáze uložena,
- **--start** – udává časový počátek ve formátu timestamp (počet sekund od data 1. 1. 1970), to jest, od kdy se začíná do databáze zapisovat. Místo timestamp lze použít zástupný znak **N,** který značí aktuální čas (now).
- **--step** – značí časový interval, ve kterém budeme do databáze data zapisovat. Výchozí hodnota je nastavena na 5 minut.
- **DS** – znamená datový zdroj následovaný jeho jménem,
- **Ds-name** – jméno datového zdroje. Lze přirovnat k názvu proměnné.
- **DST** – značí typ datového zdroje.
- **Heartbeat** – je časový údaj v sekundách, po který, nebudou-li vložena data do databáze, bude záznam označen jako neplatný.
- **Min** – značí spodní hranici měřeného intervalu, v případě, že ji neznáme, můžeme použít zástupný znak **U** jako neznámý (unknown).
- **Max** – obdoba předchozího, značící maximum intervalu.
- **CF** – konsolidační funkce značící, jak se budou data v archivech akumulovat. Muže se jednat o hodnoty AVERAGE, MIN, MAX nebo LAST.
- **xff** – xfiles factor je hodnota v procentech udávající jaká část ze vzorku může obsahovat neplatná data, aby byl celkový výsledek ještě platný.
- **steps** – jde o počet vzorků, které budou agregovány do jedné hodnoty.
- **rows** – počet řádků v RRA archivu značící velikost sledovaného období.

DST datové typy:

- **GAUGE** – používá se pro aktuální hodnoty, jako jsou teploty nebo třeba počet lidí v místnosti,
- **COUNTER** – součtové hodnoty v čase, které se nenulují, ale stále se inkrementují,
- **DERIVE** – ukládá se rozdíl dvou hodnot, které jsou od sebe vzdálené jeden step,
- **ABSOLUTE** – se používá pro hodnoty, které se při každém měření nulují.

Údaj **rows** se počítá na základě počtu vzorků, kroku a doby, po kterou chceme data v archivu uchovávat. Kupříkladu bude-li třeba uchovávat průměrná data po dobu tří týdnů (504 hodiny) v rozestupu 300 s – což je 5 minut (0,083 hodiny) po jednom vzorku na řádek, bude počet řádků, který pokrývá zadané období, roven **6 048**. Tedy 6 048 řádků, přičemž každý obsahuje data za 5 minut, což dá ve výsledku tři týdny.

Pro tento účel jsem si napsal funkci, která mi výpočet řádků zjednoduší. Jako své parametry má:

- krok v sekundách,
- velikost sledovaného intervalu v hodinách,
- počet vzorků.

Výstupem pak je řetězec ve tvaru: **vzorek:počet-řádků**. Výstup je pak použit při tvorbě RRA archivů.

```
function vratPocetZaznamu(){
  echo `awk -v krok=$1 \
  -v cas=$2 \
  -v vzorek=$3 \
  'BEGIN {print vzorek ":" sprintf("%d",(cas/(krok/60/60)))/vzorek}'`;
}
```

V práci jsem se rozhodl měřit každé 3 minuty, tj. 180 s (přičemž tento krok je volitelný v konfiguraci Monitoru), a to po maximální dobu dvou let. Volil jsem tvorbu šesti archivů RRA a vzorky, které by měly dostatečně toto období pokrýt, ale zároveň nezabrat příliš diskového prostoru. Díky tomuto rozhodnutí jsem dosáhl velikosti jedné databáze na pouhých **216 KiB.**

```
RRA:AVERAGE:0:1:40            - 2 hodiny
RRA:AVERAGE:0.5:2:120         - 12 hodin
RRA:AVERAGE:0.5:4:240         - 2 dny
RRA:AVERAGE:0.5:8:840         - 14 dnů
RRA:AVERAGE:0.5:16:1800       - 60 dnů
RRA:AVERAGE:0.5:100:3504      - 730 dnů
```

První řádek znamená, že archiv bude mít 40 řádků a v každém řádku bude průměr z jedné hodnoty. Hodnoty přibývají každé 3 minuty po jednom vzorku, potom každý řádek obsahuje hodnoty za $3 \cdot 1 = 3$ minuty. Tedy celkový časový úsek, který RRA pojme, bude $3 \cdot 40 = 120$ minut. Druhý RRA má pak 120 řádků, přičemž v každém řádku je průměr dvou hodnot získaných za čas $3 \cdot 2 = 6$ minut. Celková doba pak vychází $3 \cdot 2 \cdot 120 = 720$ minut. Třetí archív má 240 řádků a na každém je průměr čtyř hodnot získaných za čas $4 \cdot 3 \cdot 240 = 2880$ minut. Analogicky pak pro další řádky.

Hodnota **0,5** určuje v procentech, kolik ze vzorků může být neznámých, aby byl vypočtený výsledek platný. Neznámá data mohou být zapříčiněna výpadkem zařízení, proudu nebo podobně. Například u posledního RRA (100 vzorků) může být nejvíce 50 hodnot neplatných a ze zbytku se ještě vypočítá hodnota pro řádek a bude považována za platnou. V případě, že by bylo neznámých vzorků více jak 50, tak bude hodnota na tomto

řádku už prohlášena za neplatnou, což bude v grafu znázorněno jeho přerušením (**obrázek 4.8.2**) [19].

Pro tvorbu databáze a příslušných archívů RRA využívám následující kód, který je obalen ve skriptu podmínkovým blokem, kde se kontroluje, zda databáze existuje a v případě, že ano, tak se tento kód přeskočí. Databáze je vytvořena ve chvíli, kdy dojde k prvnímu testu daného zařízení, pak dochází pouze k aktualizaci dat v databázi.

```
rrdtool create "$DBDIR/$1".rrd --start N --step $PING_KROK \
  DS:odezvaMin:GAUGE:600:U:U \
  DS:odezvaAvg:GAUGE:600:U:U \
  DS:odezvaMax:GAUGE:600:U:U \
  DS:odezvaPL:GAUGE:600:U:U \
  RRA:AVERAGE:0:`vratPocetZaznamu $PING_KROK 2 1` \
  RRA:AVERAGE:0.5:`vratPocetZaznamu $PING_KROK 12 2` \
  RRA:AVERAGE:0.5:`vratPocetZaznamu $PING_KROK 48 4` \
  RRA:AVERAGE:0.5:`vratPocetZaznamu $PING_KROK 336 8` \
  RRA:AVERAGE:0.5:`vratPocetZaznamu $PING_KROK 1440 16`\
  RRA:AVERAGE:0.5:`vratPocetZaznamu $PING_KROK 17520 100`;
```

Vkládání do databáze RRD zajišťuje nástroj **rrdtool update,** který přebírá parametry ve formátu timestamp a dvojtečkami oddělený seznam vkládaných hodnot. V ping testu pak zároveň ukládám hodnoty do databáze RRD a také do SQL, kde se uloží stav (dostupný nebo nedostupný, **Obrázek 4.8.3**) a čas poslední kontroly pro přehlednost, kdy bylo zařízení naposledy kontrolováno.

```
rrdtool update "$DBDIR/$ip".rrd N:"$RRDHODNOTY";
$MYSQL -e "UPDATE Stroje SET dostupnost=1,
  kontrolovano='$START_KONTROLY' WHERE ip='$ip'";
```

4.3. Výpis trasy k cílové stanici

V průběhu konzultací bylo upřesněno zadání a byl vznesen požadavek na to, aby Monitor umožňoval přehledně zobrazit výpis jednotlivých směrovačů na cestě k cílovému zařízení. Pro tento účel jsem zvolil program **traceroute**, který je součástí všech dnes dodávaných distribucí a není ho potřeba instalovat dodatečně. O zjištění této skutečnosti se stará funkce, která poskytuje výpis ve formátu HTML, který je pak načten webovou aplikací. Funkce přebírá jeden parametr, a to adresu IP trasovaného zařízení.

```
function vypisTrasu(){
  $TRACERT $1 -n -w 1 -q 1  -m 30 | \
  grep -v -E '^traceroute' |\
  awk -vip=$1
  'BEGIN {
    print("<h1>Vypsana trasa k " ip "</h1>
    <table>\n<tr class=\"hlavicka\"><td>Hop</td><td>Cesta</td></tr>")
  }
  {
```

```
print("<tr><td>Router</td><td>" $2 "</td></tr>")
}
 END {
  print("</table>")
 }';
```

4.4. Adresářová struktura

Pro snadnější orientaci ve skriptech Monitoru jsem rozdělil patřičné části do adresářů. Adresářová struktura vypadá tak, že v rodičovském adresáři **Monitor** se nachází:

- **databaze** – adresář, ve kterém jsou uchovávány jednotlivé databáze RRD ve formátu IP_ADRESA.rrd,
- **tmp** – adresář, ve kterém jsou uloženy dočasné soubory, které využívá skript pro hledání aktivních zařízení v rozsahu, a dále pak je zde uložen log, který produkuje Monitor se zapnutou volbou **DEBUG,**
- **skripty** – adresář, ve kterém jsou uloženy užitečné skripty pro práci s Monitorem a také se tam nachází konfigurační soubor (**config.sh**), kde jsou nastaveny cesty a potřebné proměnné,
- **moduly** – adresář, ve kterém je uložen modul ping a jeho obslužné skripty.
  V případě dalšího rozvoje monitoru by bylo vhodné umísťovat další moduly právě do tohoto adresáře a dodržet určená pravidla.

4.5. Ovládání monitorovací části

Kvůli přehlednému ovládání monitorovací části jsem vytvořil jeden hlavní skript, který slouží k ovládání všech součásti Monitoru. Jednotlivé skripty je nutno pouštět právě prostřednictvím tohoto hlavního skriptu, protože samostatně je pouštět nelze. Ovládací skript nabízí přehlednou nabídku funkcí a stručnou nápovědu.

```
Nápověda pro Monitor
./monitor.sh ping
./monitor.sh graf IP
./monitor.sh trasa IP
./monitor.sh rozsah IP maska
./monitor.sh over
./monitor.sh status
./monitor.sh verzovac
```

Parametry:

- **ping** – získá z SQL databáze IP adresy a nad každou provede test dostupnosti. Po dokončení testu jsou výsledky uloženy do RRD databáze.

- **graf IP** – generuje grafické náhledy pro zadanou adresu ve formátu png a nakopíruje do adresáře **obrazky,** který se nachází ve webové části Monitoru. Absolutní cesta k webovému adresáři je uvedena v konfiguračním souboru-scripty/config.sh.
- **trasa IP** – vypíše trasu k zadanému cíli a to ve formátu HTML. Tohoto výstupu využívá pak webová část monitoru.
- **rozsah IP maska (bitový zápis)** – zjistí, které adresy jsou ze zadaného intervalu (síť a maska) aktivní, a ty zařadí do sledování.
- **over** – pomocná volba, která zjistí předpoklady pro běh Monitoru a snaží se uživatele navést k odstranění chyb nebo doinstalovaní potřebných balíčků.
- **status** – pomocná volba, která vypíše, kolik místa na pevném disku zabírá adresář s obrázky a adresář s databázemi RRD.
- **verzovac** – vývojová volba, která promaže dočasné soubory a zabalí veškeré skripty do jednoho souboru. **Tato volba nezálohuje data samotná, ale pouze skripty**. Struktura databáze SQL zálohována je, protože je součástí pomocných skriptů. Použití doporučuji v případě, že se v kódech budou dělat nějaké úpravy nebo rozšíření. Jako unikátní název souboru používá timestamp, aby bylo možné jednotlivé verze od sebe jednoznačně odlišit.

4.6. Spouštění Monitoru

Aby bylo možné provádět testy dostupnosti pravidelně, bylo nutné nějakým způsobem vyřešit periodické spouštění monitorovacího skriptu.

Nabízely se dvě možnosti, a to jednak napsat **nekonečnou smyčku**, ve které bude příkaz čekat požadovaný časový úsek a spouštět monitorovací skript nebo použít nástroj přímo k tomu určený, a to **cron.**

U první možnosti by bylo nutné zajistit spouštění smyčky při startu počítače, a to mi jako řešení nepřišlo příliš efektivní. Rozhodl jsem se tedy využít crontab a spouštět monitorování v něm. Příkaz, který přidá záznam do crontabu a bude periodicky každé tři minuty spouštět monitorování, vypadá následovně:

```
crontab -e
*/3     *       *       *       *       /cesta/Monitor/monitor.sh ping
```

Problém, se kterým jsem se setkal při běhu monitorovacích skriptů, které byly volány skrze funkce PHP, byl problém s právy. Je třeba si uvědomit, že skripty jsou volány pod uživatelem, stejným jako běží web-server. Na GNU/Debian je to uživatel www-data, a

ten musí mít přístup do adresářové struktury Monitoru**.** Monitorovací skripty, ale obvykle poběží pod jiným uživatelem. Typicky pod tím, který bude monitorování provozovat. Jeden ze způsobů je **nastavit práva** uživateli, pod kterým běží web server, www-data do této struktury, a to jak pro čtení, tak pro zápis (do určitých adresářů a souborů). Nebo přijít na způsob, jak toto obejít. Možnost, která se nabízí je zajistit **běh skriptů volaných skrze PHP pod jejich vlastníkem.** To umožňuje modul do webového serveru **suPHP**, který volá nejprve PHP interpret a jako parametr mu předává volaný PHP skript. Toto řešení je jednak bezpečnější, protože není potřeba povolovat jiným uživatelům přístup do svého adresáře, a jednak to zvyšuje přehlednost provozu Monitoru, protože vše běží pod jedním uživatelem. Vzhledem k různým systémům a konfiguracím, na kterých může být Monitor provozován, jsem v návodu na instalaci níže popsal oba způsoby s tím, že použití suPHP bych doporučoval.

4.7. Zabezpečení monitorovací části

Vzhledem k tomu, že skripty přebírají většinou argumenty ve formátu adresy IP, bylo nutné nějakým způsobem zajistit to, aby použitím chybného argumentu nedošlo k neočekávanému chování. Vytvořil jsem funkci, která kontroluje formát adresy IP a v případě úspěchu vrací 0 jinak 1.

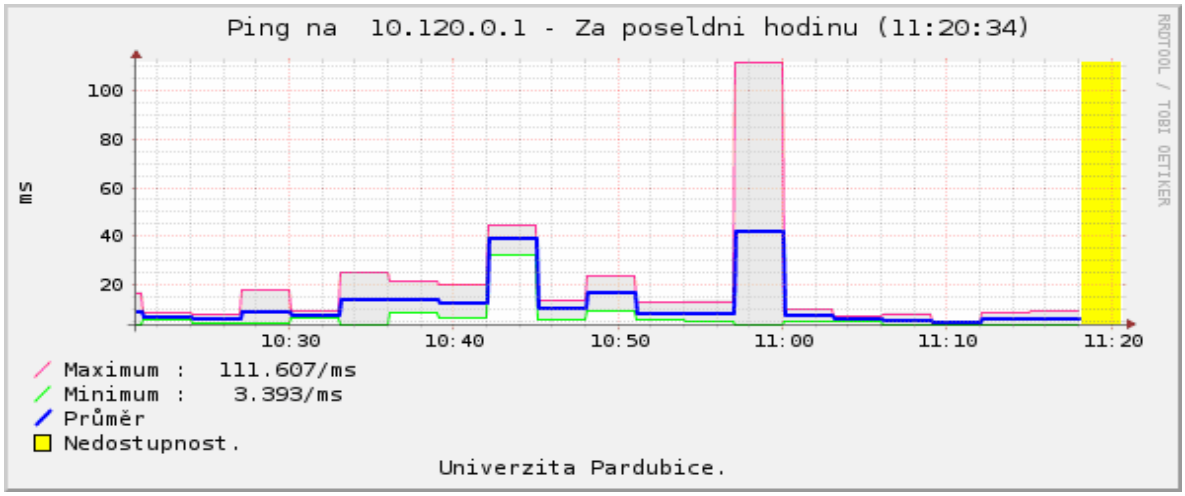
```
function jeIP(){
  echo $1 |\
  awk ' {
    uspech=0;
    split($0,IP,/\./);
      for(i in IP){
        if( (IP[i]>=0) && (IP[i])<=255){
          uspech+=1;
        }
      }

      if(uspech==4){
        print 0;
      } else {
        print 1;
      }
  }';
};
```

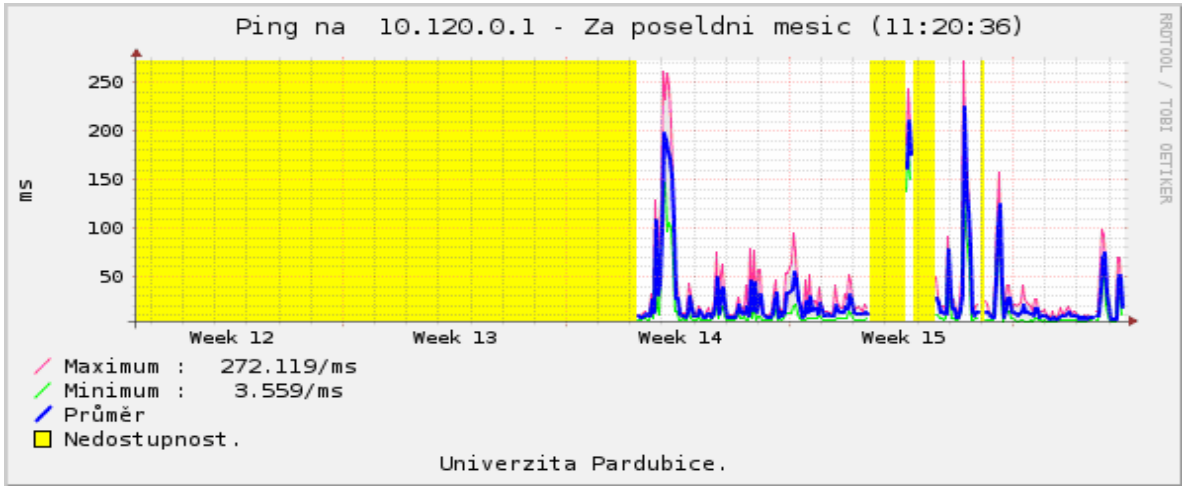
4.8. Grafická prezentace

Pro grafickou prezentaci naměřených dat jsem se rozhodl využít formu grafu v kombinaci s barevnou signalizací (**Obrázek 4.8.3**) znázorňující dostupnost nebo nedostupnost daného zařízení.

Grafy jsou kresleny od jedné hodiny až po jeden rok. Maximální doba, za kterou je možné zobrazit graf, je dva roky. Přičemž je v grafech znázorněn pouze rok. Velikost kresleného intervalu lze ovlivnit, ovšem nikoliv na uživatelské úrovni pomocí konfigurace, ale zásahem do skriptu, který daná období vykresluje (moduly/ping/vytvorGraf.sh) a upravit v něm požadované rozsahy dle potřeb. V případě, že chceme vykreslit graf za delší období než dva roky, tak je potřeba ještě upravit rozsahy RRA archivů (moduly/ping/ping.sh). Myslím si ale, že grafy za období jednoho roku jsou dostačující. V grafu jsou vykresleny křivky pro maximální, průměrnou a minimální odezvu daného zařízení. Nedostupnost je v grafu zvýrazněna a signalizována přerušením grafu v době nedostupnosti, jak je dobře vidět na **Obrázek 4.8.2**. Grafická prezentace vypadá pak následovně:



Obrázek 4.8.1 – Hodinový graf dostupnosti



Obrázek 4.8.2 – Měsíční graf dostupnosti

10.120.4.6	Trasa	Nedefinovan	Default Mr.	ANO	2010-04-22 11:21:01	ANO	Upravit
~~10.120.4.9~~	~~Trasa~~	~~Nedefinovan~~	~~Default Mr.~~	~~ANO~~	~~2010-04-12 23:33:02~~	~~NE~~	~~Upravit~~
10.120.4.10	Trasa	Nedefinovan	Default Mr.	ANO	2010-04-22 11:21:01	ANO	Upravit
10.120.4.13	Trasa	Nedefinovan	Default Mr.	ANO	2010-04-22 11:21:01	ANO	Upravit
10.120.4.14	Trasa	Nedefinovan	Default Mr.	NE	2010-04-22 11:21:01	ANO	Upravit
10.120.4.17	Trasa	Nedefinovan	Default Mr.	ANO	2010-04-22 11:21:01	ANO	Upravit

Obrázek 4.8.3 – Přehled značení stavu zařízení

4.8.1. Generování obrazových náhledů

Jako výstupní formát pro obrázky grafu jsem volil PNG, který je pro použití na webu velmi vhodný a často používaný. Důvodem k výběru byla velikost výsledných obrazových souborů a také fakt, že není vázán patenty, jako tomu bylo například u formátu GIF.

Původní myšlenka byla, že bych generoval grafy přímo po provedení testu dostupnosti, ovšem to by znamenalo navýšení běhu testovacího skriptu o zhruba **dvě sekundy**. Tak dlouho totiž trvá vygenerování šesti obrazových náhledů (na použitém systému). Tento čas by šel pravděpodobně snížit výkonnějším hardwarem. Tento způsob se neukazuje jako ideální řešení, protože to nejen zatěžuje monitorovací systém, ale také prodlužuje běh testu.

Většinou není potřeba generovat všechny grafy, ale pouze některé a uživatelem vyžádané.

Jako další možnost se nabízelo použít v databázi u konkrétního zařízení příznak, který by zajistil generování při dalším testu. Přičemž by musel uživatel jednak zatrhnout příznak, a jednak čekat do dalšího provedení testu, který by na základě vyhodnocení příznaku generoval obrázky. Tento způsob sice šetří systémové prostředky, ale zase na druhou stranu prodlužuje dobu čekání uživatele na výsledný graf. I toto řešení se mi jevilo jako nevhodné.

Třetí možnost jak generovat obrazové náhledy byla ta, že by uživatel klikl na odkaz a tato událost by daný graf vytvořila a načetla. Toto řešení nabízí jednak šetření systémových prostředků, protože jsou generovány grafy pouze v tu chvíli, kdy je uživatel potřebuje, a jednak odpadá i čas na kontrolu příznaku a čekání na další periodu. Jediné zpoždění, které lze pozorovat, je právě výše uvedený čas kolem dvou sekund, který zabere vygenerování obrazových náhledů.

Při implementaci jsem se rozhodl použít třetí popsanou možnost, jejíž výhody jsou popsány výše v textu. Funkce, která se stará o generování obrázků přebírá tři parametry a to interval, ve kterém chceme graf vytvořit, popisek grafu a textový řetězec, který je použit v textu grafu. Funkce vypisuje ve formátu HTML příslušné obrázky a vypadá následovně:

```
function vytvorGraf(){
$RRDTOOL graph "$IMGDIR/$IP"_"$3".png \
  --width 500 \
  --height 150 \
  --start $1 \
  --end N \
  --v "ms" \
  -E \
  --rigid \
  --alt-autoscale \
  DEF:min=$DB.rrd:odezvaMin:AVERAGE \
  DEF:max=$DB.rrd:odezvaMax:AVERAGE \
  DEF:avg=$DB.rrd:odezvaAvg:AVERAGE \
  CDEF:nedostupnost=avg,UN,INF,UNKN,IF \
  AREA:max#EEEEEE:"" \
  LINE1:max#FF3e96:"Maximum \\:" \
  GPRINT:max:MAX:"%6.3lf/ms \\n" \
  AREA:min#FFFFFF:"" \
  LINE1:min#00EE00:"Minimum \\:" \
  GPRINT:min:MIN:"%6.3lf/ms \\n" \
  LINE2:avg#0000ff:"Průměr  \\n" \
  AREA:nedostupnost#ffff00:"Nedostupnost.\\n" \
  COMMENT:"FEI Univerzita Pardubice. \\c" \
  --title "Ping na  $IP - $2 (`date +%T`)"  \
  --imginfo    "<IMG    SRC="\"obrazky/%s\""    WIDTH="%lu"    HEIGHT="%lu"
ALT="\"Graf\"">";
}
```

Další funkce využívá předešlé funkce pro generování HTML výstupu k obrázkům a generuje soubor PHP, který obsahuje odkazy pro všech šesti obrázků, a pak je načítán webovou částí Monitoru.

```
function vytvorPHPDokument(){
echo "<div class="center">";
  vytvorGraf "now-1h"         "Za poseldni hodinu"          "hodina" ;
  vytvorGraf "now-6h"         "Za poseldnich 6 hodin"       "6hodin" ;
  vytvorGraf "now-1d"         "Za poseldni den"             "den" ;
  vytvorGraf "now-7d"         "Za poseldni tyden"           "tyden" ;
  vytvorGraf "now-1month"     "Za poseldni mesic"           "mesic" ;
  vytvorGraf "now-1y"         "Za poseldni rok"             "rok" ;
echo "</div>";
} > $IMGDIR/$IP.php;
```

4.8.2. Zobrazení výsledků

Pro zobrazení výsledku a přívětivější ovládání jsem volil aplikaci PHP/HTML, která umožňuje snadné zadávání adres a rozsahů IP a také snadnou úpravu či vyřazení jednot-

livých elementů ze sledování. PHP má výbornou podporu databáze MySQL, takže mi toto spojení přišlo jako výhodné, když jsou data uložená právě v MySQL.

V PHP aplikaci jsem se snažil odsunout kód pracující s databází do samostatných tříd, protože to zvyšuje přehlednost a usnadňuje úpravy v kódu, a tedy využít alespoň částečně objektově orientovaný přístup. Úplná objektová podpora, jako je tomu například v jazyce JAVA v PHP chybí. Snažil jsem se tedy využít alespoň částečně toho, co PHP nabízí. V aplikaci jsou použity dvě stěžejní třídy.

- **Stroje** – tato třída zajišťuje práci jako je vkládání, zobrazení a mazání sledovaných prvků.
- **Vlastníci** – tato třída zajišťujíce práci s daty nad tabulkou Vlastníci, která je zde uvedena spíše jako demonstrace napojení monitoru na nějakou již existující databázi, jak bylo poznamenáno v popisu SQL struktury výše.

4.9. Zabezpečení webové aplikace

Vzhledem k tomu, že vkládám data skrze webové formuláře, která jsou předávána přímo do dotazů SQL, může dojít k podstrčení škodlivých dat a to jak úmyslně, tak i nevědomky. Bylo nutné nalézt způsob, jak tomu zabránit, protože následky zákeřného kódu by mohly být pro aplikaci destruktivní. Tento typ útoku je nazýván **SQL injection**. Volil jsem tedy vlastní funkci, která kontroluje povolené znaky při vkládání, a v případě výskytu nepovoleného znaku, je znak z vkládaného řetězce odstraněn. Funkci využívají obě výše zmíněné třídy ke kontrole formulářových dat. Výstupem je pak řetězec zbavený nepovolených znaků.

```
protected static function kontrolaVstupu(&$vstup){
        $nepovoleneZnaky = array('?',';','\'', '"', '(', ')', '!', '#',
'$','%','^','&','*','-','+','=','`','\\','~','/','>','<');
        return str_replace($nepovoleneZnaky,'',$vstup);
    }
```

5. Instalace Monitoru

5.1. Požadavky

Pro to, aby bylo možné Monitor správně provozovat, je nutné na systému, kde bude monitorování nasazeno, nainstalovat následující softwarové vybavení:

- balíček rrdtool,
- program nmap,
- server MySQL a jeho konzolového klienta,
- web server s podporou PHP, MySQL,
- modul web-serveru suPHP (doporučení).

Na systému GNU/Debian je možno nainstalovat balíčky příkazem:

```
aptitude install apache2 php5 mysql-server mysql-client rrdtool nmap
php5-mysql
```

5.2. Průvodce v krocích

Výrazy, které mají na začátku a na konci podtržítko, jsou dále ve výkladu myšleny jako proměnné.

5.2.1. Nastavení databáze SQL

Nejprve je nutné vytvořit uživatele s heslem a příslušnou databázi v SQL serveru (MySQL).

*Vytvoření uživatele **rudolf** s heslem **delnik** a možnost připojovat se pouze z lokálního počítače.*
```
mysql -uroot -p_HESLO_ -e "CREATE USER 'rudolf'@'localhost' IDENTIFIED BY
'delnik'";
```

*Vytvoření databáze **monitor***
```
mysql -uroot -p_HESLO_ -e "CREATE DATABASE monitor";
```

*Nastavení přístupu do databáze **monitor** pro uživatele **rudolf***
```
mysql -uroot -p_HESLO_ -e "GRANT ALL PRIVILEGES ON monitor.* to 'rudol-
f'@'localhost' WITH GRANT OPTION ";
```

5.2.2. Vytvoření struktury databáze SQL

Vytvoření struktury databáze můžeme učinit pomocí skriptu **scripty/databaze.sql**. V případě, že se nacházíme v adresáři **Monitor,** tak následujícími příkazy:

```
cd scripty;
mysql -urudolf -pdelnik monitor < databaze.sql;
```

5.2.3. Kopírování složek

Monitor se skládá ze **dvou** komponent, a to z monitorovací a webové části. Je nutné nakopírovat adresáře **Monitor** (adresář s monitorovacími skripty) a **MonitorWEB** (adresář s webovou částí) na potřebné místo.

*Nakopírování adresáře **Monitor** do adresáře **/home/rudolf/***
```
/home/rudolf/Monitor # Absolutní cesta
```

*Nakopírování adresáře **MonitorWeb** do adresáře **/var/www/***
```
/var/www/MonitorWeb # Absolutní cesta
```

5.2.4. Proměnné

Nastavení potřebných proměnných v souboru scripty/config.sh. Uvedené hodnoty proměnných jsou předpokládány z předchozích kroků výše.

V sekci "Údaje pro databázi SQL"
```
USER="rudolf";
PASS="delnik";
HOST="localhost";
DATABAZE="monitor";
```

*Úprava konfigurace webové části **MonitorWeb/config.php***
```
$HOST="localhost";
$USER="rudolf";
$PASS="delnik";
$DB="monitor";
$monitor="/home/rudolf/Monitor"; // Což je cesta k monitorovacím skrip-
tům.
```

*Nastavení proměnné **WWWDIR** (absolutní cesta k adresáři s webem) v sekci "Cesty k adresářům".*
```
WWWDIR="/var/www/MonitorWeb" # bez lomítka na konci!
```

*Nastavení proměnné **ROOT** v ./ monitor.sh v části konfigurace na absolutní cestu k adresáři Monitor.*
```
ROOT="/home/rudolf/Monitor" # Bez lomítka na konci!
```

5.2.5. Ověření předpokladů

Ověření předpokladů dosáhneme pomocí předpřipraveného skriptu.

```
/home/rudolf/Monitor/./monitor.sh over
```

Pokud příkaz vypíše **vše v pořádku,** tak upravíme záznam v cronu a přidáme do něj periodické spouštění skriptů každé tři minuty. Tuto volbu lze ovlivnit v konfiguraci modulu ping scripty/config.sh **KROK_PING**="POČET SEKUND". Ve většině případů, ale není nutné měnit. Snad jen při velkém počtu sledovaných prvků, kdy by se běh skriptů překrýval s dalším spuštěním.

Při testu **168** zařízení, z čehož bylo **13** nedostupných, běh testu trval **81 s**; což odpovídá skutečnosti, že test nedostupného zařízení trvá jednu sekundu a test dostupného

0,5 s; tedy $77{,}5 + 13 = 91$ sekund. Chybějících 10 s je způsobeno režií při ukládání dat do RRD a SQL databáze.

```
crontab -e
*/3   *     *     *     *     /home/rudolf/Monitor/monitr.sh ping
```

5.2.6. Konfigurace suPHP modulu

- Vypnout mod_php(!)
- instalovat php5-cgi,
- instalovat libapache2-mod-suphp.

Nastaveni modulu apache
```
/etc/apache2/mods-enabled/suphp.conf
  <IfModule mod_suphp.c>
        suPHP_Engine on
        AddType application/x-httpd-suphp .php
        suPHP_AddHandler application/x-httpd-suphp
  </IfModule>
```

Nastaveni suphp
```
/etc/suphp/suphp.conf
check_vhost_docroot=false
```

Restart apache
```
/etc/apache2/init.d/apche2 restart
```

5.2.7. Úprava práv

Úprava práv (tento krok je možné přeskočit v případě použití modulu Apache serveru **suPHP**) pro uživatele www-data (pod kterým běží standardně PHP skripty na GNU/Debian). Uživatel musí mít možnost zapisovat do adresáře **tmp** a i do souborů s možností přepisovat soubory. V případě, že nebude fungovat funkce „Projdi rozsah“, je to tím, že www-data nemůže zapisovat do **tmp/dostupneIP.txt**.

```
chmod -R go+rwx /var/www/Monitor/obrazky
chmod -R go+rwx /home/rudolf/Monitor/tmp
```

5.2.8. Doporučení

V praxi se mi osvědčilo, že když něco nefunguje, jsou na vině práva. Uživatel, pod kterým běží web server musí mít přistup do adresáře Monitor:

Pro čtení
```
Monitor/databaze # Odkud čte data pro generování grafů.
Monitor/moduly/ping # Odkud spouští potřebné skripty které musí mít flag
+x executable.
Monitor/scripty #Odkud pouští potřebné skripty
```

Pro zápis
```
Monitor/tmp # Kam si ukládá aktivní IP adresy z rozsahu.
MonitorWeb/obrazky # Kam zapisuje vygenerované obrázky.
```

Je dobré, si funkčnost ověřit tak, že se můžeme přepnout ze super uživatele root na uživatele www-data, a zkusíme si přístup do adresářů. Nebo další možností je příkaz **ls** a **chmod** pro nastavení práv.

V případě, že spustíme k testovacím účelům „**./monitor.sh graf IPů“**, tak to vytvoří obrázky a soubory s právy uživatele, pod kterým se skript spustil. Přičemž pak nemůže uživatel www-data tato data přepsat při generování nových grafů, pokud neupravíme práva.

Pro možnost detailnějších výpisů k ověření, zda skripty fungují, je potřeba v souboru **scripty/config.sh** správně zapnout volbu **DEBUG=1**. Potřebné logy pak najdeme v **tmp/monitr.log**. Doporučuji volbu po čase testování vypnout, protože soubor s logem může zabírat při delším běhu hodně místa.

Závěr

Mým cílem v této bakalářské práci bylo vybrat vhodnou technologii pro monitoring směrovačů a na základě této technologie implementovat vlastní řešení, což se mi dle mého úsudku povedlo splnit výběrem protokolu ICMP.

Vzhledem k tomu, že jsem se s podobnou situací (tvorba monitorovacího systému) ještě nesetkal, tak jsem narazil během implementace na pár problémů, které se mi podařilo úspěšně vyřešit.

Prvním problémem byla volba vhodného softwarového vybavení, které jsem volil na základě svých vlastních zkušeností. Cílem byla minimalizace obslužných programů. Důvodem využití minima nástrojů byla snaha o multiplatformnost, kterou jsem ověřit nemohl, protože jsem měl k dispozici pouze zařízení se systémem GNU/Linux. Portace by však vyžadovala pár úprav ve zdrojovém kódu.

Dalším problémem byla volba dat, která by vystihla co nejlépe sledovaný stav a poskytla uživateli užitečné informace vedoucí k odhalení možných problémů na lince. Rozhodl jsem se tedy sledovat čtyř veličiny, ze kterých lze odvodit možné problémy vyskytující se na lince.

Cíl, který jsem si stanovil, byla minimalizace zátěže prvků, kde se bude sledování realizovat. Myslím, že jsem cíle dosáhl jednak vhodnou volbou nástrojů, a jednak toho, že generování grafů probíhá pouze na vyžádání uživatele a zbytečně tak nezatěžuje systém. Díky využití nástrojů RRD se mi podařilo minimalizovat i nároky na diskový prostor.

V programu jsem se snažil připravit možnosti pro další rozšíření. Jako další rozšíření se nabízí například sledování vytíženosti centrálního prvku, nebo sledování vzdálených stanic. To by ale znamenalo použití protokolu SNMP, čemuž jsem se při sledování dostupnosti vyhnul. Jako další rozšíření je možné sledování průtoku dat. Tyto úpravy by znamenaly zásah do obou částí Monitoru, ale vzhledem ke snaze o objektový přístup by to nemělo působit potíže.

Úprava, která by byla vhodná přímo u modulu ping, je vytvořit strom cest k danému cíli a následně ho pak využít při zobrazování stavů, kde by při výpadku kořene byly jeho listy znázorněny jako nedosažitelné nikoli však nedostupné jako je tomu teď, protože jejich status nelze díky výpadku kořene ověřit.

Oproti původnímu zadání jsem rozšířil aplikaci ještě o možnost vypisovat trasy ke sledovanému cíli.

Ovládání a zobrazení naměřených stavů jsem volil webovou aplikaci a grafy, z nichž lze naměřené údaje přehledně a snadno přečíst. Možnost náhledu na historii umožňuje porovnat vývoj daných veličin a na základě toho pak odhalit vznikající problém.

Téma této bakalářské práce se mi skutečně zdálo velmi zajímavé a přimělo mě to nastudovat řadu efektivních nástrojů a různých odborných článků. Myslím si, že rozšíření aplikace by mohlo být velmi přínosné, a tím tak docílit komplexního monitorovacího systému.

Seznam ilustrací

Seznam tabulek

Seznam zkratek a pojmů

AWK – Aho, Wieneberger, Kerningham – programovací jazyk

BASH – Bourn Against Shell – příkazový interpret

GNU – GNU's Not Unix – rekurzivní akronym, projekt zabývající se tvorbou svobodného softwaru

HTML – HyperText Markup Langure – značkovací jazyk sloužící k prezentaci informací na internetu

ICMP – Internet Controlol Message Protokol – servisní protokol používaný v IP protokolu

IP – Internet Protokol – protokol užívaný k adresaci uzlu internetové sítě

MIB – Management Informatik Base – databáze uchovávající stavy a informace o daném zařízení

MySQL – databázový server

OID – Object Identifier – jednoznačný identifikátor elementu v databázi MIB

OOP – Object Oriented Programing – metodika vývoje softwaru

PHP – Hypertext Preprocesor – programovací jazyk používaný k tvorbě dynamických stránek HTML

PNG – Portable Network Graphics – formát obrázku hojně užívaný při webové prezentaci

PostreSQL – databázový server

RRA – Round Robin Archive, obdoba tabulky v SQL, ve které jsou uloženy časové údaje za určité časové období

RRD – Round Robin Diabase, způsob uchovávání a agregace dat

SNMP – Standard Network Management Protokol – protokol pro správu, síťových prvků

SQL – Structured Query Langure – dotazovací jazyk používaný v relačních databázích

TCP – Transmission Control Protocol – protokol spojového charakteru zajišťující bezpečný přenos sítí

UDP – User Datagram Protocol – protokol zajišťující nespojový přenos

Zdroje

1. ICMP In *Wikipedia: the free encyclopedia* [online]. St. Petersburg (Florida): Wikipedia Foundation, 2010?, [cit. 2010-04-27]. Dostupné z WWW: <http://cs.wikipedia.org/wiki/ICMP>.
2. BOUŠKA, Petr. *Začínáme s monitoringem sítě* [online]. 2009 [cit. 2010-04-27]. Dostupné z WWW: <http://www.samuraj-cz.com/clanek/zaciname-s-monitoringem-site/>.
3. ODVÁRKA, Petr. *ICMP – Internet Control Message Protocol* [online]. 2001 [cit. 2010-04-27]. Dostupné z WWW: <http://www.svetsiti.cz/view.asp?rubrika=Technologie&clanekID=36>.
4. Traceroute In *Wikipedia: the free encyclopedia* [online]. St. Petersburg (Florida): Wikipedia Foundation, 2010?, [cit. 2010-04-27]. Dostupné z WWW: <http://cs.wikipedia.org/wiki/Traceroute>.
5. *Linux Documentation* [online]. 2009? [cit. 2010-04-27]. Ping(8) – Linux man page. Dostupné z WWW: <http://linux.die.net/man/8/ping>.
6. *Linux Documentation* [online]. 2009? [cit. 2010-04-27]. Traceroute(8) – Linux man page. Dostupné z WWW: <http://linux.die.net/man/8/traceroute>.
7. SPORTACK, Mark A. *Směrování v sítích IP : Autorizovaný výukový průvodce*. Brno : Computer Press, 2004. 391 s.
8. KABELOVÁ, Alena; DOSTÁLEK, Libor . *Velký průvodce protokoly TCP/IP a systémem DNS*. Praha : Computer Press, 2008. 488 s.
9. MRÁZEK, Oldřich. *SNMP protokol a jeho využití* [online]. 2003 [cit. 2010-04-27]. Dostupné z WWW: <http://hw.cz/ethernet/snmp/vyuziti_snmp.html>.
10. BOUŠKA, Petr. *SNMP – Simple Network Management Protoco.* [online]. 2006 [cit. 2010-04-27]. Dostupné z WWW: <http://www.samuraj-cz.com/clanek/snmp-simple-network-management-protocol/>.
11. FERBAS, Dušan. *Protokol SNMP v průmyslových zařízeních* [online]. 2004 [cit. 2010-04-28]. Dostupné z WWW: <http://hw.cz/Produkty/Ethernet/ART1094-Protokol-SNMP-v-prumyslovych-zarizenich.html>.
12. KLAŠKA, Luboš. *Model Manager – Agent* [online]. 200 [cit. 2010-04-28]. Dostupné z WWW: <http://www.svetsiti.cz/view.asp?rubrika=Tutorialy&temaID=23&clanekID=30>.

13. PUŽMANOVÁ, Rita. *Moderní komunikačni´sítě od A do Z*. Praha : Computer Press, 1998. 446 s.
14. KLAŠKA, Luboš. *Principy RMON* [online]. 2000 [cit. 2010-04-28]. Dostupné z WWW: <http://www.svetsiti.cz/view.asp?rubrika=Tutorialy&temaID=23&clanekID=34>.
15. KLAŠKA, Luboš. *Základní skupiny RMON* [online]. 2000 [cit. 2010-04-28]. Dostupné z WWW: <http://www.svetsiti.cz/view.asp?rubrika=Tutorialy&temaID=23&clanekID=36>.
16. RMON In *Wikipedia: the free encyclopedia* [online]. St. Petersburg (Florida): Wikipedia Foundation, 2010?, [cit. 2010-04-27]. Dostupné z WWW: <http://en.wikipedia.org/wiki/RMON>.
17. Miscellaneous Functions. *17. MySQL: Developer Zone* [online]. 2009 [cit. 2010-04-29]. Dostupný z WWW: <http://dev.mysql.com/doc/refman/5.0/en/miscellaneous-functions.html#function_inet-aton>.
18. RRDTool In *AbcLinuxu.cz – Linux na stříbrném podnose* [online]. Praha (Česká Republika): Stickfish, 2009 [cit. 2010-04-28]. Dostupné z WWW: <http://www.abclinuxu.cz/software/system/monitorovani/rrd>.
19. Co mi uklada RRD databaza? In *AbcLinuxu.cz - Linux na stříbrném podnose* [online]. Praha (Česká Republika): Stickfish, 2009 [cit. 2010-04-28]. Dostupné z WWW: <http://www.abclinuxu.cz/poradna/linux/show/252163>.