VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV INFORMAČNÍCH SYSTÉMŮ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF INFORMATION SYSTEMS

# VYUŽITÍ ČASOVÝCH INFORMACÍ PRO IDENTIFIKACI POČÍTAČE

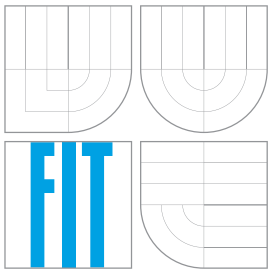
DIPLOMOVÁ PRÁCE
MASTER'S THESIS

AUTOR PRÁCE
AUTHOR

Bc. JAKUB JIRÁSEK

BRNO 2012

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV INFORMAČNÍCH SYSTÉMŮ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF INFORMATION SYSTEMS

# VYUŽITÍ ČASOVÝCH INFORMACÍ PRO IDENTIFIKACI POČÍTAČE

COMPUTER IDENTIFICATION USING TIME INFORMATION

DIPLOMOVÁ PRÁCE
MASTER'S THESIS

AUTOR PRÁCE
AUTHOR

Bc. JAKUB JIRÁSEK

VEDOUCÍ PRÁCE
SUPERVISOR

Ing. LIBOR POLČÁK

BRNO 2012

# Abstrakt

Tato práce se zabývá problematikou vzdálené identifikace počítače. Využívá k tomu časová razítka TCP získaná od sledovaného počítače. Z těchto razítek je možné určit posun vnitřních hodin sledovaného počítače. Tento posun je pro každý počítač jedinečný. Výsledek identifikace není ovlivněn lokací sledovaného počítače, síťovou adresou ani způsobem připojení. Díky využití pasivního odposlechu je tato identifikace pro sledovaný počítač neviditelná. Předpokladem úspěšného rozpoznání je možnost sledujícího počítače zachytit síťovou komunikaci od sledovaného počítače. Tato komunikace musí probíhat nad protokolem TCP a pakety musí ve svých hlavičkách obsahovat časová razítka.



# Abstract

This work deals with the identification of a remote computer by monitoring TCP timestamps of the tracked device. It is possible to determine computer's clock skew from these timestamps as the clock skew is unique for every device. We are able to differentiate devices even though they have changed location, network address or connection type. Passive data capturing ensures that the identification process is invisible to the fingerprinted computer. It is necessary that the network communication of fingerprinted computer is visible to the observing device. We are able to utilise only TCP traffic with timestamps enabled.



# Klíčová slova

Identifikace počítače, vnitřní hodiny, posun hodin, časová razítka.

# Keywords

Computer identification, device's clock, clock skew, timestamps.


# Citace

Jakub Jirásek: Využití časových informací pro
identifikaci počítače, diplomová práce, Brno, FIT VUT v Brně, 2012

# Využití časových informací pro identifikaci počítače

## Prohlášení

Prohlašuji, že jsem tuto diplomovou práci vypracoval samostatně pod vedením pana Ing. Libora Polčáka. Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

.....
Jakub Jirásek
21. května 2012

## Poděkování

Na tomto místě bych rád poděkoval vedoucímu této diplomové práce, panu Ing. Liborovi Polčákovi, za cenné rady, náměty a připomínky týkající se práce a panu Ing. Matěji Grégrovi, za pomoc při praktickém nasazování a testování implementovaného nástroje.

# Obsah

# Seznam obrázků

# Kapitola 1

# Úvod

Jednou z velkých výhod Internetu je pro mnoho lidí jistá míra anonymity. Na rozdíl od reálného světa zde mohou lidé poměrně jednoduše vystupovat pod jinou identitou, čehož pak nejčastěji využívají na různých fórech a sociálních sítích. Problém však nastává v případě, kdy za pomoci počítače dojde k útoku na vzdálený systém. Na řadu pak přichází forenzní analýza za účelem vypátrání pachatele. Základem úspěchu je získání síťové adresy počítače útočníka. Existuje však mnoho způsobů, jak tuto adresu změnit. Od jednoduchého dynamického přidělovaní adres od poskytovatele připojení, přes ukrytí počítače za směrovač (NAT), až po využití různých více či méně veřejných proxy serverů či anonymizačních sítí. Postup, jak počítač útočníka vypátrat, může být náročný a zdlouhavý.

Pokud se však přeci jen podaří útočníkův počítač nalézt, k prokázání viny či neviny by nám mohl pomoci nástroj, který by dokázal jednoznačně rozhodnout, zda se skutečně jedná o stejný počítač, ze kterého byl útok proveden, a to i v případě, že již tento počítač neobsahuje žádná data.

Vytvoření nástroje, který dokáže rozpoznat a identifikovat počítač, je náplní této diplomové práce. Výše zmíněné využití však nemusí být zdaleka jediné. Nástroj může být součástí vzdálené autorizace svázané s konkrétním počítačem, pomocníkem při lokalizaci přenosného zařízení nebo může sloužit k identifikaci jednoho počítače vystupujícího pod různými adresami.

Tato práce si tak klade za cíl posunout dál myšlenku zjišťování užitečných informací o vzdáleném počítači na úroveň jeho jednoznačné identifikace. Rozpoznání je možné díky drobnému posunu vnitřních hodin sledovaného počítače, který je pro každé zařízení jedinečný. K tomu nám postačuje síťová komunikace od sledovaného počítače nad protokolem TCP. Nezáleží přitom na tom, kde se počítač nachází, jakou má síťovou adresu nebo zda je připojený za směrovačem a využívá překlad síťových adres (NAT).

Tímto tématem se již dříve zabývali Tadayoshi Kohno, Andre Broido a K.C. Claffy [8]. Tato práce se snaží navázat na jejich výzkum a implementovat funkční nástroj schopný pracovat v reálném síťovém prostředí. Nedílnou součástí práce tak bylo zkoumání a experimentování se současným hardwarovým vybavením a moderními operačními systémy.

První část kapitoly 2 slouží k uvedení do problematiky identifikace počítače a je zde vysvětlen pojem vnitřních hodin počítače. Hlavní náplní kapitoly je popis různých možností získání časových informací od sledovaného počítače a jejich vzájemné srovnání.

Kapitola 3 vysvětluje princip výpočtu posunu vnitřních hodin ze získaných časových razítek TCP. Poskytuje tak ucelený postup získání výsledného posunu včetně vysvětlení použitého názvosloví. Na závěr kapitoly jsou uvedeny další dvě možnosti využití získaných časových razítek TCP.

Náplní kapitoly 4 je popis implementovaného nástroje. Jsou zde vysvětleny použité technologie a princip práce programu.

Popis provedených experimentů a dosažených výsledků je součástí kapitoly 5. V této části práce jsou také uvedeny aspekty a problémy využití implementovaného nástroje v reálném prostředí.

Poslední kapitola 6 obsahuje shrnutí a zhodnocení dosažených výsledků a navazující vývoj projektu.

Práce vychází ze Semestrálního projektu, ve kterém byly z velké části zpracovány kapitoly 2, 3 a část návrhu z kapitoly 4.

# Kapitola 2

# Získání časových informací

Všechna síťová zařízení (servery, směrovače, klienti...) mají vlastní vnitřní hodiny sestávající z hardwarových a softwarových komponent. Takt těchto hodin udává oscilátor řízený krystalem, tikajícím vždy na určité frekvenci. Deklarovaná frekvence však nikdy není úplně přesná, vliv na ni mají okolní teplota, vlhkost a především typ samotného krystalu [21]. Počet taktů oscilátoru počítá čítač.

Z vnitřních hodin počítače je nastavován i systémový čas. Ten může být následně korigován, nejčastěji pomocí protokolů NTP [12] či SNTP [11].

Jak již bylo zmíněno v úvodu, použitý způsob jednoznačné identifikace počítače spočívá v určení posunu vnitřních hodin tohoto počítače. Jak ale hodinový takt zjistit vzdáleně, pouze ze síťového provozu? Existuje několik možností a na ty se podíváme v rámci této kapitoly. U každé z možností jsou navíc zmíněny výhody a nevýhody použití dané metody k jednoznačné identifikaci počítače. Ještě před tím si však uvedeme informace o synchronizaci času počítače pomocí protokolu NTP a rozdíl mezi pasivním a aktivním odposlechem dat.

## 2.1 Časová synchronizace protokolem NTP

Z důvodu nepřesnosti vnitřních hodin počítačů byl vytvořen protokol NTP, jehož čtvrtá verze je popsána v RFC 5905 [12]. Protokol je navržený pro distribuci časových informací pomocí paketů UDP nad protokolem IP. Časové informace klientům poskytují NTP servery, vhodnými algoritmy je pak i přes různá časová zpoždění jednotlivých klientů dosahováno přesnosti v řádu několika málo milisekund[1].

Základní požadavky, které musí být při časové synchronizaci protokolem NTP splněny [13]:

- Hodiny primárních NTP serverů musí být buď synchronizovány podle národního standardu, nebo se jedná o zkalibrované atomové hodiny.
- Poskytované časové údaje musí být přesné, bez ohledu na umístění cílového počítače.
- Synchronizace musí být spolehlivá i přes možné výpadky spojení (redundance serverů, různé cesty).
- Protokol musí pracovat nepřetržitě a poskytovat časové informace dostatečně často na to, aby u cílového počítače kompenzoval posun jeho hodin.

[1] Přesnost je značně závislá na době, po jakou je synchronizace aktivní.

- Systém musí pracovat nad existující sítí, s minimálními požadavky na cílový počítač a jeho operační systém.

Synchronizace hodin může pracovat ve dvou režimech — démon a jednorázová synchronizace. Účelem jednorázové synchronizace je v konkrétním okamžiku nastavit správný čas. V případě démona NTP navíc dochází ke korekci posunu vnitřních hodin počítače. Vliv a praktický dopad časové synchronizace na identifikaci počítačů je rozebrán v kapitole 5.

## 2.2 Pasivní versus aktivní odposlech dat

Abychom byli vůbec schopni od vzdáleného počítače nějaké časové informace získat, musí v první řadě mezi oběma zařízeními existovat síťové spojení. Dále v této kapitole budeme předpokládat, že již takové spojení existuje. Jak daleko (resp. přes kolik dalších síťových zařízení) jsou sledovaný a sledující počítač od sebe vzdálené, pro nás tady není důležité.

Odposlech dat může být buď pasivní nebo aktivní. Při pasivním odposlechu musí klient sám od sebe komunikovat s naslouchajícím počítačem. Může se jednat například o

- webový server, se kterým klient často komunikuje,
- bránu poskytovatele internetového připojení,
- internetovou službu vyžadující častou aktualizaci, například předpověď počasí či zpravodajství.

Jednoznačnou výhodou tohoto typu odposlechu je fakt, že je pro sledovaný počítač neviditelný.

Při aktivním odposlechu nás také zajímají přijatá data, jejich poslání si však sami vynutíme. Většinou odešleme na stranu klienta dotaz a počkáme na odpověď. Jedná se například o zprávy ICMP (viz kapitola 2.3) či požadavky HTTP, kde webový server vystupuje jako sledovaný počítač. I když se nemusí jednat o nic podezřelého, inicializací a aktivním zapojením se do komunikace přestává být sledující počítač neviditelný.

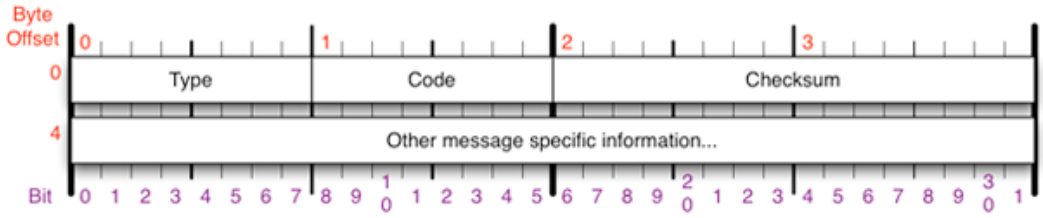
Navíc existuje tzv. semi-pasivní technika [8], při které s klientem aktivně komunikujeme, ale až poté, co sám komunikaci zahájil. Příkladem může být síťová služba udržující spojení s klientem, ale až poté, co se klient ke službě sám přihlásil.

## 2.3 Časová razítka ICMP

*Internet Control Message Protokol* (ICMP) je součástí protokolu IP [18] a slouží primárně k posílání chybových zpráv při zpracování datagramů IP a pro směrovací účely. Zpráva ICMP je tak poslána například pokud datagram nemůže dosáhnout požadovaného cíle nebo nese informaci o nové adrese pro směrování datagramů (RFC 792 [17]).

O jakou zprávu ICMP se jedná nám určuje její typ — prvních 8 bitů datagramu (viz obrázek 2.1). Tato hodnota nám také říká, v jakém formátu jsou zbývající data. RFC 792 [17] definuje 11 různých typů zpráv ICMP a pro nás jsou důležité typy s čísly 13 a 14. Tyto zprávy totiž obsahují časová razítka. Kód zprávy je u těchto dvou typů nastaven na 0, není potřeba přenášet žádné doplňující informace.

Tělo zprávy ICMP typu 13 a 14 obsahuje tři 32-bitová čísla, vyjadřující počet milisekund uběhnutých od půlnoci. *Originate timestamp* vyplní odesílatel svým časem, příjemce pak odpoví dvěma svými časy: *Receive timestamp* a *Transmit timestamp*. Rozdíl mezi těmito




Obrázek 2.1: Zpráva ICMP

časovými razítky je ten, že *Recieve timestamp* vyplní systém časem, kdy zprávu přijal, a *Transmit timestamp* časem, kdy zprávu odeslal[2] [17]. Navíc může tělo zprávy obsahovat *identifikaci* a *sekvenční číslo* pro zajištění integrity.

Tento způsob je velice podobný metodě využívající časová razítka TCP (podkapitola 2.6), na rozdíl od ní má však dvě nevýhody:

- Zkoumaný počítač nesmí zprávy ICMP filtrovat, což dnes většina firewallů dělá[3] [8].
- Pro získání datagramů ICMP je nutná aktivní účast odposlouchávacího nástroje.

Naopak výhodou může být pevně stanovená frekvence zvyšování časového razítka (1 kHz, viz RFC 792 [17]), odpadá tedy nutnost tento údaj počítat. Vzhledem k výše zmíněným nevýhodám se však tato metoda pro identifikaci počítače příliš nehodí.

## 2.4 Časová razítka HTTP

Využitím časových razítek HTTP pro jednoznačnou identifikaci — v tomto případě webových serverů — se zabývali Sebastian Zander a Steven J. Murdoch [21]. Jejich cílem bylo odhalení služeb běžících na počítačích skrytých za anonymizační sítí. Příkladem takové anonymizační sítě může být Tor[4], nabízející uživatelům skrytí identity včetně přenášených dat. Práce dále ukazuje, jak může být využití časových razítek HTTP výhodné, především co se týče rychlosti a množství přenesených dat.

Základní myšlenka vychází z faktu, že součástí datové hlavičky odpovědi webového serveru na požadavek HTTP od klienta je také datum a čas. Pro HTTP 1.0 [2] je to chování doporučené, pro HTTP 1.1 [4] je to chování povinné.

Časový údaj je inkrementován po sekundách, jeho frekvence je tedy 1 Hz. Takto nízká frekvence se už na první pohled jeví jako značné nevýhodná pro měření drobných odchylek, většinou v řádu mikrosekund. Hlavním důvodem je tzv. *chyba kvantování*, která může být v nejhorším případě téměř jedna sekunda a způsobené kolísání odchylek znemožní jakákoliv měření v kratších intervalech.

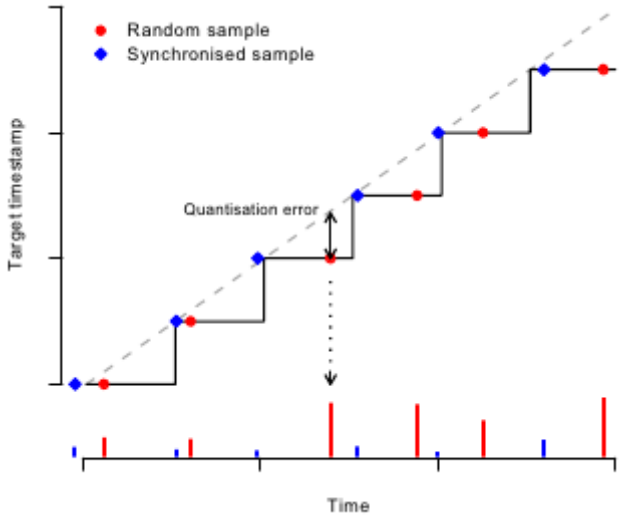
Co znamená chyba kvantování je vidět na obrázku 2.2. Na x-ové ose je reálný čas, na y-ové jsou hodiny sledovaného počítače, ze kterých generuje časová razítka. Přerušovanou čárou je znázorněna funkce, vyjadřující lineární růst hodin sledovaného počítače v závislosti na čase. Protože jsou však časová razítka generována po sekundách, má reálná funkce tvar schodovitý. Na obrázku je reálná funkce zobrazena plnou čárou. Chyba kvantování (angl.

[2]Při experimentování s těmito zprávami byla tato časová razítka vždy shodná, což ale pro naše účely ničemu nevadí.

[3]Nemusí se vždy jednat pouze o firewall běžící na koncové stanici, zprávy ICMP může filtrovat i směrovač/brána, přes kterou komunikace prochází.

[4]WWW stránky: Tor. `https://www.torproject.org/`.

*quantisation error*) v určitém časovém okamžiku je pak vzdálenost této reálné funkce od funkce lineární, viz obrázek 2.2. Obrázek také ukazuje možné řešení v podobě synchronizace obou porovnávaných hodin. Červeně jsou znázorněna časová razítka přijatá v náhodných časových okamžicích, modře pak časová razítka v přesných časových momentech za účelem eliminace chyby kvantování. Díky tomuto postupu tak lze měřit i drobné odchylky vnitřních hodin sledovaného počítače a to s relativně malým počtem přijatých dat [21].




Obrázek 2.2: Výhoda synchronizovaných hodin oproti náhodně zaslaným dotazům. (Obrázek převzat z [21].)

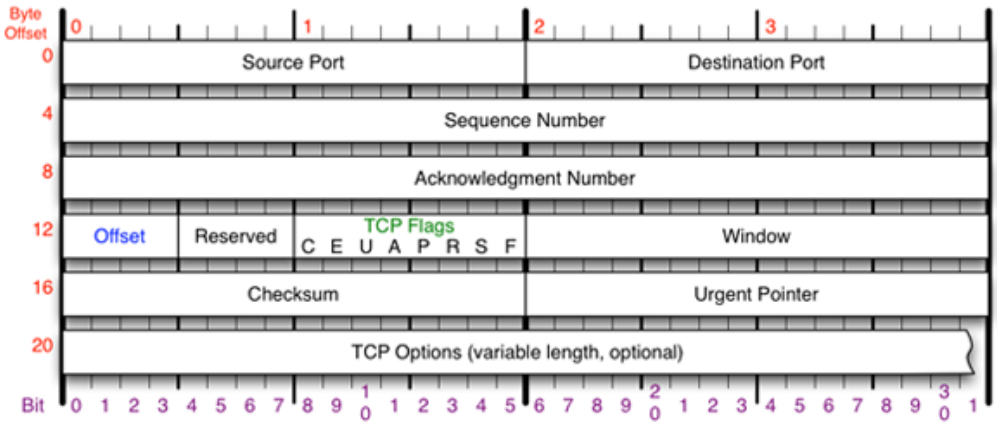
Použití běžného času v sekundách pro identifikaci počítače je poměrně velkou výzvou, pro naše účely se však nehodí. Jednak je tato metoda značné omezena pouze na webové servery a navíc vyžaduje plně aktivní účast identifikačního nástroje.

## 2.5 Sekvenční čísla TCP

Součástí každé hlavičky TCP je sekvenční číslo (angl. *sequence number*, obrázek 2.3). Tato 32-bitová hodnota slouží pro určení správného pořadí paketů u příjemce a k potvrzování jejich správného doručení. Zajišťuje tak protokolu TCP spolehlivost a zaručuje doručení všech paketů. Odesílatel vždy čeká na potvrzení o doručení paketu a pokud potvrzení včas nedostane, odešle paket znovu. K jednoznačné identifikaci paketů pak slouží právě sekvenční čísla.

Popis sekvenčních čísel a vlastností, které musí splňovat, je popsán v RFC 793 [19]. Tři nejdůležitější vlastnosti jsou tyto:

- Hodnota sekvenčního čísla prvího paketu v rámci jednoho spojení je číslo nejmenší, u dalších paketů pak hodnoty postupně rostou.
- Inicializační sekvenční čísla v rámci různých spojení TCP musí být dostatečně rozdílná, aby bylo jednoznačně rozlišitelné, ke kterému spojení patří.
- Inicializační sekvenční číslo musí být vybráno takovým způsobem, aby bylo těžko odhadnutelné pro třetí stranu.




Obrázek 2.3: Hlavička TCP

Přesná pravidla pro generování inicializačních sekvenčních čísel však RFC 793 nespecifikuje. Implementace u různých operačních systémů se tak značně liší a z důvodu bezpečnosti její popis nenajdeme ani ve veřejně dostupné literatuře [16]. S. J. Murdoch a S. Lewis však zjistili [16], že operační systém Linux používá ke generování inicializačních sekvenčních čísel, mimo jiné, také časový údaj.

I v rámci různých verzí operačního systému Linux se implementace generování inicializačních sekvenčních čísel mírně odlišují. Co však zůstává stejné, je přidání časového údaje jako nejvýznamnějšího bajtu generovaného oktetu. Díky tomu je tak zachována rostoucí tendence těchto čísel. Konkrétně u verze jádra 2.6 se jedná o čas v sekundách od startu systému, podělený konstantou 300. K této hodnotě je pak přidán systémový čas v mikrosekundách [16].

Hlavní výhodou využití sekvenčních čísel jako zdroje časových informací od vzdáleného počítače je skutečnost, že je tato položka povinnou součástí každé hlavičky TCP, což by zajišťovalo využití téměř veškerých přijatých dat. Čas v mikrosekundách a tedy frekvence 1 MHz se také jeví jako velice výhodné pro měření drobných odchylek. Hlavním problémem je tak omezení pouze na operační systém Linux.

## 2.6 Časová razítka TCP

Jako nejvhodnější zdroj časových informací pro vzdálenou identifikace počítače se jeví *TCP timestamp*. Je součástí hlavičky TCP, je generován z vnitřních hodin počítače a pouhým pasivním odposlechem umožňuje získání velkého množství informací z komunikace nad protokolem TCP. Jako nevýhoda se může jevit rozdílná frekvence u různých operačních systémů (mezi 1 Hz a 1 kHz [21]), její zjištění však není nijak složité. Jedinou opravdovou nevýhodou tak zůstává fakt, že pole *TCP timestamp* je volitelné a není striktně určeno, jakým způsobem musí být používáno. K čemu slouží a jak s ním může být nakládáno popisuje RFC 1323 [7].

RFC 793 [19], popisující protokol TCP, se o poli *TCP Options* (viz obrázek 2.3) v rámci hlavičky TCP vyjadřuje velice skromně. Vše, co se dočteme o jeho využití, je, že má sloužit především pro testovací účely.

RFC 1323 [7] se pak zabývá rozšířením TCP pro zlepšení výkonnosti, převážně z důvodu vyšších přenosových rychlostí. Používá k tomu právě pole *TCP Options*.

Využití časových razítek v rámci hlavičky TCP je pak dvojí [7]:

**RTTM** (Z angličtiny — *Round Trip Time Measurement*). RTT, neboli čas uplynutý od odeslání paketu po přijetí jeho doručení, je poměrně zásadní pro určování velikosti tzv. *receive window* — maximálního počtu paketů odeslaných bez zatím přijatých potvrzení. Díky znalosti RTT můžeme rychleji reagovat na změny, protože dopředu víme, kdy nám jednotlivá potvrzení mají přijít. Jedním z využití *TCP timestamp* je tak měření RTT. Odesílatel přidá do hlavičky TCP své časové razítko, které mu příjemce v odpovědi (paket s nastaveným ACK příznakem) vrátí. Po přijetí potvrzení tak odesílatel jednoduše zjistí, jaká doba uplynula od odeslání paketu po přijetí potvrzení.

**PAWS** (Z angličtiny — *Protect Against Wrapped Sequence Numbers*). PAWS mechanismus je ochrana proti akceptování paketu z předešlého spojení TCP. V jednoduchosti funguje asi takto — příjemce se podívá na hodnotu časového razítka a pokud je příliš stará, jedná se o paket z předešlého spojení a zahodí ho.

*TCP timestamp* tvoří dvě 32-bitová čisla. První z nich (*TSval*) obsahuje časové razítko vygenerované odesílatelem při odesílání paketu, druhé (*TSecr*) je součástí potvrzovacího paketu (paketu s příznakem ACK) a je vyplněno časovým razítkem posledního doručeného segmentu. Důležitým poznatkem pro naše účely je, že je hodnota časového razítka — poslaná jako *TSval* — získána z hodin, které jsou alespoň přibližně stejné jako reálný čas [7]. Pro většinu dnešních implementací to znamená vzít hodnotu z vnitřních hodin počítače.

# Kapitola 3

# Princip identifikace počítače

Co jsou to vnitřní hodiny počítače a jakým způsobem fungují, již bylo zmíněno v úvodu kapitoly 2. Tam byla také uvedena pro nás velmi důležitá informace, že v porovnání s reálným časem nejsou tyto hodiny vždy úplně přesné. To v praxi znamená drobný, ale přesto měřitelný posun vnitřních hodin počítače od reálného času. Tento posun vnitřních hodin je pak mezi různými počítači dostatečně velký na to, aby je bylo možné jednoznačně rozlišit. Cílem této kapitoly je poskytnout čtenáři ucelený přehled informací potřebných pro pochopení principu jednoznačné idenifikace počítače na základě časových informací získaných z časových razítek.

Na závěr této kapitoly si představíme další dvě možnosti využití časových razítek TCP, konkrétně identifikaci operačního systému a odhadnutí času startu systému sledovaného počítače. Jedná se o doplnění informací o praktickém využití získaných časových razítek.

## 3.1 Odchylka a posun vnitřních hodin počítače

V následující podkapitole použijeme terminologii zavedenou v [8]. Mějme hodiny C, reprezentující čas uběhnutý od počátečního času i[C]. Hodnota r[C] je nejmenší časová jednotka, o kterou může být čas inkrementován. Frekvence hodin, Hz[C], ja pak inverze r[C]

$$Hz[C] = \frac{1}{r[C]}, \tag{3.1}$$

platí tedy (například)

$$r[C] = 10ms \Rightarrow Hz[C] = 100Hz. \tag{3.2}$$

Nechť $t$ udává reálný čas a R[C]$(t)$ značí čas udaný hodinami C v čase $t$. Odchylkou hodin C, off[C], budeme značit rozdíl mezi časem udaným hodinami C a reálným časem $t$

$$\forall t \geq i[C]\ .\ off[C](t) = R[C](t) - t. \tag{3.3}$$

Posun hodin, s[C], je první derivace odchylky hodin podle času, přičemž předpokládáme, že R[C] je diferencovatelná (spojitá) funkce v $t$. Jednotkou posunu hodin jsou buď mikrosekundy za sekundu ($\mu s/s$) nebo počet částí v milionu (*ppm*, z angl. *parts per million*) [8, 14].

Aby dále v textu nedocházelo k záměně pojmů *odchylka* a *posun* hodin, zde je ještě jednou shrnuto použité názvosloví:

**odchylka hodin** - angl. *offset*, rozdíl časových údajů porovnávaných hodin, s časem buď lineárně roste nebo klesá, [off[C]] = $ms$, $s$

**posun hodin** - angl. *skew*, posun porovnávaných hodin, konstantní s časem, [s[C]] = $\mu s/s$, *ppm*

Na posun vnitřních hodin se také můžeme dívat jako na rychlost změny odchylky. Využití znalosti této hodnoty k jednoznačné identifikaci počítače je možné díky faktu, že v rámci jednoho počítače tento posun leží v rozmezí $\pm(1-2)$ ppm, zatímco v rámci různých počítačů je rozdíl až $\pm 50$ ppm [15].

## 3.2 Určení posunu hodin z časových razítek

I v rámci této podkapitoly se budeme z větší části držet terminologie z [8]. Předpokládejme, že již máme od sledovaného počítače získanou množinu paketů TCP, obsahujících časová razítka podle doporučení RFC 1323 (viz podkapitola 2.6). Z této množiny vytvoříme posloupnost, seřazenou podle času přijetí jednotlivých paketů. Tuto posloupnost nazveme $T$. Nechť $t_i$ je čas v sekundách, kdy sledující počítač zachytil $i$-tý paket z posloupnosti $T$. Platí tedy

$$\forall i \in \{1, \ldots, |T|\} \, . \, t_i \geq t_{i-1}. \tag{3.4}$$

Při popisu budeme obecně předpokládat, že čas $t_i$ je čas reálný. Protože však i na sledující straně stojí počítač, nemusí se v praxi vždy jednat o reálný čas, ale o čas u sledujícího počítače. Na obecnosti řešení se tím mění fakt, že naměřené hodnoty, včetně výsledného posunu vnitřních hodin, nejsou hodnoty absolutní, ale vždy relativní vzhledem k času sledujícího počítače.

Proměnná $x_i$ nám určuje relativní čas v sekundách, uběhnutý od počátku měření po doručení $i$-tého paketu

$$x_i = t_i - t_1. \tag{3.5}$$

Druhý důležitý parametr pro určení posunu hodin vzdáleného počítače nazveme $v_i$. Jedná se o rozdíl hodnoty časového razítka $i$-tého paketu $T_i$ a hodnoty časového razítka paketu prvního, $T_1$

$$v_i = T_i - T_1. \tag{3.6}$$

Jak $x_i$, tak $v_i$ nám určují časový údaj. Abychom s nimi však mohli pracovat současně, musí mít také stejný rozměr. Zatímco $x_i$ nám udává čas v sekundách, u $v_i$ je hodnota závislá na frekvenci $f$ hodin C sledovaného počítače. Před jejím použitím ji tak převedeme na sekundy

$$w_i = v_i/f. \tag{3.7}$$

Pokud frekvenci dopředu neznáme, musíme ji vypočítat. Frekvenci získáme jako derivaci hodnot časových razítek podle času. Protože výsledná hodnota nebude téměř nikdy úplně přesná, zaokrouhlíme ji na nejbližší celé číslo

$$f = \frac{\mathrm{d}T}{\mathrm{d}t}[Hz]. \tag{3.8}$$

Odchylka off[C] $i$-tého paketu je rozdíl mezi časem udaným časovým razítkem (který je zároveň odrazem vnitřních hodin sledovaného počítače) a časem reálným. Jeho hodnota může být jak kladná (hodiny jdou napřed), tak záporná (hodiny se opožďují)

$$y_i = w_i - x_i. \tag{3.9}$$

Výstupem výše zmíněných výpočtů je potom posloupnost odchylek $O_T$, náležící k získané posloupnosti paketů $T$ a seřazená podle času přijetí

$$O_T = ((x_i, y_i) : i \in \{1, \ldots, |T|\}). \tag{3.10}$$

Dvě takové posloupnosti odchylek $O_T$ jsou zobrazeny na obrázku 3.1. Poslední otázkou zůstává, jak z posloupnosti odchylek určit posun vnitřních hodin sledovaného počítače.

### 3.2.1 Metoda založená na lineárním programování

Sue B. Moon, Paul Skelly a Don Towsley navrhli [14] pro určení posunu vnitřních hodin počítače využít metodu založenou na lineárním programování. Vychází z faktu, že je posun dvou porovnávaných hodin s časem konstantní. Zpoždění jednotlivých paketů se sice může měnit (dojde k zahlcení linky, jinému směrování apod.), při dostatečně dlouhém měření je ale právě lineární změna naměřených odchylek známkou konstantního posunu obou hodin. To je patrné i při pohledu na graf odchylek (obrázek 3.1). Chceme-li zjistit rychlost změny odchylek, můžeme proložit body přímkou a změřit úhel, který tato přímka svírá s osou $x$.

K proložení grafu je použita lineární funkce, shora ohraničující množinu všech odchylek[1]. Výsledná hodnota tak není zatížena chybami způsobenými zpožděnými pakety a kopíruje nejrelevantnější hodnoty

$$\alpha \cdot x_i + \beta \geq y_i, i \in \{1, \ldots, |T|\} \tag{3.11}$$

Druhou důležitou vlastnost, kterou funkce musí splňovat, je minimální vertikální vzdálenost všech bodů posloupnosti odchylek $O_T$ od této funkce

$$min\{\sum_{i=1}^{|T|}(\alpha \cdot x_i + \beta - y_i)\}. \tag{3.12}$$

Hledaný posun vnitřních hodin sledovaného počítače je pak hodnota $\alpha$, určující sklon této lineární funkce. Na obrázku 3.1 je zobrazena modrou přerušovanou čárou.

### 3.2.2 Metoda nejmenších čtverců

Druhou možností [9], jak určit lineární posun porovnávaných hodin, je proložit body grafu (naměřené odchylky) přímkou, která k daným bodům „nejlépe pasuje“. K nalezení takové přímky je možné využít metodu nejmenších čtverců. Tato metoda hledá pro danou množinu bodů přímku

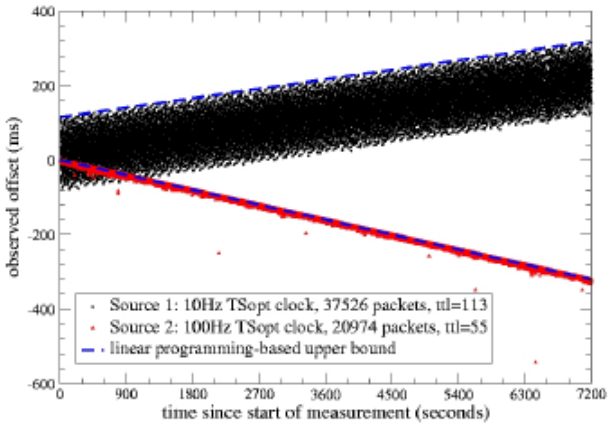
$$y_i = \alpha \cdot x_i + \beta, i \in \{1, \ldots, |T|\} \tag{3.13}$$

takovou, že součet druhých mocnin vertikálních vzdáleností všech bodů je od této přímky nejmenší

$$min\{\sum_{i=1}^{|T|}(\alpha \cdot x_i + \beta - y_i)^2\}. \tag{3.14}$$

Výsledná přímka však může být značně ovlivněna body s velkou odchylkou. V našem případě by se jednalo o body znázorňující zachycené pakety, které dorazily s velkým zpožděním.

[1] V [14] je namísto množiny odchylek jednotlivých paketů použita množina zpoždění, hledaná funkce proto ohraničuje body zdola. Na principu metody to však nic nemění.

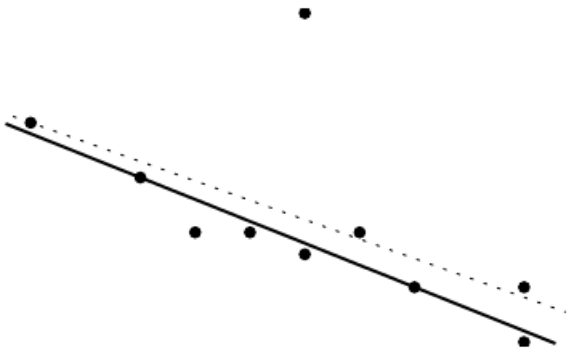



Obrázek 3.1: Zobrazení dvou posloupností odchylek časových razítek TCP. Černou barvou jsou vyznačena časová razítka od počítače s frekvencí hodin 10 Hz, červenou od počítače s frekvencí 100 Hz. Modrými přerušovanými čárami jsou graficky zobrazeny hledané posuny vnitřních hodin u obou sledovaných počítačů, nalezené pomocí lineárního programování. (Obrázek převzat z [8].)

Součtem absolutních hodnot odchylek namísto jejich druhých mocnin můžeme tento vliv zmírnit

$$min\{\sum_{i=1}^{|T|}|\alpha\cdot x_i+\beta-y_i|\}. \tag{3.15}$$

Tato metoda je velice podobná metodě předcházející a je také založena na principu lineárního programování. Vždy však bude, více či méně, ovlivněna zpožděnými pakety. Ty mají pro určení posunu sledovaných hodin nulovou vypovídací hodnotu a negativně tak budou ovlivňovat výsledek. Pro naše účely je výhodnější vybrat pakety s co nejmenším zpožděním, což odpovídá právě přímce shora ohraničující všechny získané hodnoty.



Obrázek 3.2: Proložení bodů přímkou. Tečkovanou čárou je zobrazena přímka získaná metodou nejmenších čtverců, plnou čárou přímka získaná součtem absolutních hodnot. (Obrázek převzat z [9].)

## 3.3 Další možnosti využití časových razítek TCP

Časová razítka TCP se dají využít i k dalším dvěma účelům — mohou pomoci s identifikací běžícího operačního systému a často se z nich dá vyčíst čas startu systému. To jsou ostatně dva hlavní důvody, proč administrátoři časová razítka TCP u svých systémů zakazují.

### 3.3.1 Identifikace operačního systému

I když je návrh a způsob použití komunikačních protokolů většinou striktně popsán v daném RFC, mohou na určité události reagovat jiné operační systémy různě. Jedná se například o předem nedefinovaný či nepovolený způsob použití. Jedním z příkladů může být zaslání paketu se současným nastavením příznaků SYN a RST. Je pak na operačním systému, jak se s takovým paketem vypořádá. Rozdíly mohou být nejen v rámci různých operačních systémů, ale i v rámci jiných verzí stejného systému [20].

Pro identifikaci operačních systémů existuje celá řada nástrojů, z těch nejznámějších jmenujme například: `nmap`[2], `Xprobe`[3] (oba aktivní odposlech) či `p0f`[4] (pasivní odposlech).

Program `nmap` pro detekci operačního systému posílá až 16 paketů TCP, UDP a ICMP, speciálně upravených tak, aby zužitkovaly nejednoznačnosti vzniklé při definici daného protokolu. Z odpovědí si pak nástroj vytváří tzv. otisk (angl. *fingerprint*), který následně porovnává s otisky uloženými v databázi. Podle míry shody pak dokáže s danou pravděpodobností určit operační systém včetně jeho verze.

Tabulka 3.1: Frekvence zvyšování časového razítka TCP u různých operačních systémů.

| **Operační systém** | **Frekvence [Hz]** |
|---|---|
| Windows XP SP3 | 10 |
| Windows 7 | 100 |
| OpenSolaris 2009.06 | 100 |
| Android 2.3.7 | 100 |
| Debian Linux 5.0 (verze jádra 2.6.26) | 250 |
| Arch Linux (verze jádra 3.2.8) | 300 |
| CentOS Linux 5.5 (verze jádra 2.6.32) | 1000 |
| FreeBSD 9.0 | 1000 |

Jednou z metod, kterou `nmap` používá pro identifikaci operačního systému, je i zjišťování frekvence, s jakou jsou navyšovány hodnoty časového razítka v paketech TCP (časová razítka TCP jsou popsána v podkapitole 2.6). Tato frekvence se totiž u různých operačních systémů může lišit a to v rozmezí 1 Hz až 1 kHz (RFC 1323 [7], sekce 4.2.2). Frekvence zvyšování časových razítek u různých operačních systémů jsou uvedeny v tabulce 3.1. Čistě znalost této frekvence sice k identifikaci operačního systému nestačí, její znalost však může pomoci s omezením množiny možných výsledků a je proto součástí zjišťování operačního systému programem `nmap`.

Příklad použití programu `nmap` k identifikaci operačního systému (FreeBSD 8.2-STABLE):

```
nmap -O 147.229.176.xxx
```

[2] WWW stránky: Nmap security scanner. `http://nmap.org/`.
[3] WWW stránky: X probe - active OS fingerprinting tool. `http://sourceforge.net/projects/xprobe/`.
[4] WWW stránky: p0f. `http://lcamtuf.coredump.cx/p0f.shtml`.

```
Starting Nmap 5.51 ( http://nmap.org ) at 2011-12-06 14:20 CET
Nmap scan report for 147.229.176.xxx
Host is up (0.028s latency).
Not shown: 967 closed ports
PORT      STATE    SERVICE

...

Device type: general purpose
Running: FreeBSD 7.X|8.X
OS details: FreeBSD 7.0-RELEASE-p1 - 8.1-RELEASE-p1
Network Distance: 9 hops

OS detection performed. Please report any incorrect results at
http://nmap.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 18.35 seconds
```

### 3.3.2 Zjištění času startu systému

U většiny operačních systémů můžeme z časového razítka TCP odvodit také čas startu systému, respektive dobu, po kterou daný systém běží (angl. *uptime*). V žádném z RFC není specifikováno, jakým způsobem má být vybráno první (inicializační) časové razítko, dnešní operační systémy tak často při svém startu nastaví čítač časových razítek na nulu. U takovéhoto systému nám pak k určení doby běhu systému stačí znát frekvenci generování časových razítek a jednu aktuální hodnotu časového razítka [10]. U všech systémů uvedených v tabulce 3.1 bylo možné čas startu určit.

# Kapitola 4

# Návrh a implementace nástroje

Praktickým výstupem práce je nástroj, schopný na základě sledování síťového provozu identifikovat počítače. Implementovaný nástroj s názvem `pcf` (z angl. *pc-fingerprinter*) je ve formě démona, naslouchajícím na síťovém rozhraní. Podle nastaveného filtrování program sleduje síťový provoz a analyzuje veškerou komunikaci. Výstupem a současně komunikačním kanálem s dalšími nástroji je soubor XML, který obsahuje seznam sledovaných zařízení. Ze souboru je možné vyčíst všechny důležité informace o jednotlivých počítačích, včetně naměřeného posunu vnitřních hodin a případné identifikace počítače s některým z uložených či právě sledovaných zařízení. Databáze pro uložení počítačů je také ve formě souboru XML, jejíž formát je velice podobný seznamu sledovaných zařízení. Ke každému z právě sledovaných počítačů je navíc automaticky generován graf jeho odchylek a naměřeného posunu, což umožňuje sledovat vývoj jednotlivých měření v grafické podobě.

Nadstavbou nad implementovaným nástrojem je webové rozhraní, umožňující pracovat s nástrojem pomocí internetového prohlížeče. Rozhraní umožňuje sledovat všechna aktivní měření (včetně jednotlivých grafů), zaznamenat rozpoznaná zařízení a pracovat s databází uložených počítačů.

Samotný nástroj je implementován v jazyce C a je primárně určen pro operační systém Linux.

## 4.1 Základní schéma programu

Základní schéma programu `pcf` je zobrazeno na obrázku 4.1. Po úvodní inicializaci se program dostane do čekací rutiny na nově přijatý paket, pakety jsou pak tříděny a ukádány podle zdrojové síťové adresy. Všechny výpočty jsou prováděny po blocích, do získání potřebného počtu paketů se jednotlivé pakety (resp. časová razítka TCP) pouze ukládají do paměti. Pokud došlo k přijetí dostatečného množství paketů od některého ze sledovaných počítačů, dojde u daného počítače k výpočtu frekvence zvyšování hodnoty jeho časových razítek, určení odchylek jednotlivých paketů a k výpočtu posunu jeho vnitřních hodin. Pomocí nově získaného posunu dojde k pokusu o identifikaci zařízení, vygenerování grafu a uložení změn do souboru. Běh programu se pak vrací zpět do čekací rutiny na nový paket. K ukončení programu dojde buď po uplynutí zvolené doby, získáním daného počtu paketů nebo přijetím jednoho ze signálů SIGINT/SIGTERM.

Na základě tohoto návrhu je implementace programu rozdělena do pěti modulů:

- načtení konfigurace,
- odposlech dat,

- zpracování přijatých dat a generování grafů,
- matematické výpočty,
- identifikace počítačů a ukádání výsledků do souboru XML.

## 4.2 Odposlech dat

Základem nástroje `pcf` je sledování síťového provozu na zvoleném rozhraní. K tomuto účelu je využita knihovna pcap, konkrétně její implementace *libpcap*[1], která je primárně určena pro unixové operační systémy (Linux, Solaris, BSD...) a je šířena jako svobodný software pod licencí BSD.

Knihovna *libpcap* pracuje na úrovni jádra operačního systému. Díky tomu má přímý přístup k veškerým datům procházejícím přes síťové rozhraní, včetně údaje o přesném čase zachycení paketu. Tento časový údaj je v mikrosekundách a uložen je ve struktuře `timeval` z hlavičkového souboru `<sys/time.h>`. Další výhodou je možnost nastavení filtrování paketů, pracující na úrovni jádra operačního systému. Z důvodu přímého přístupu k síťovému rozhraní jsou pro správnou funkci knihovny požadována administrátorská práva, tato práva tak budou potřeba i při spouštění implementovaného nástroje [5].

## 4.3 Zpracování přijatých dat

Prvotní zpracování dat probíhá již při filtrování paketů pomocí knihovny *libpcap*. Základem je filtr pro odmítnutí jiných paketů než paketů TCP spolu s odfiltrováním paketů, jejichž hlavička TCP zjevně neobsahuje časová razítka:

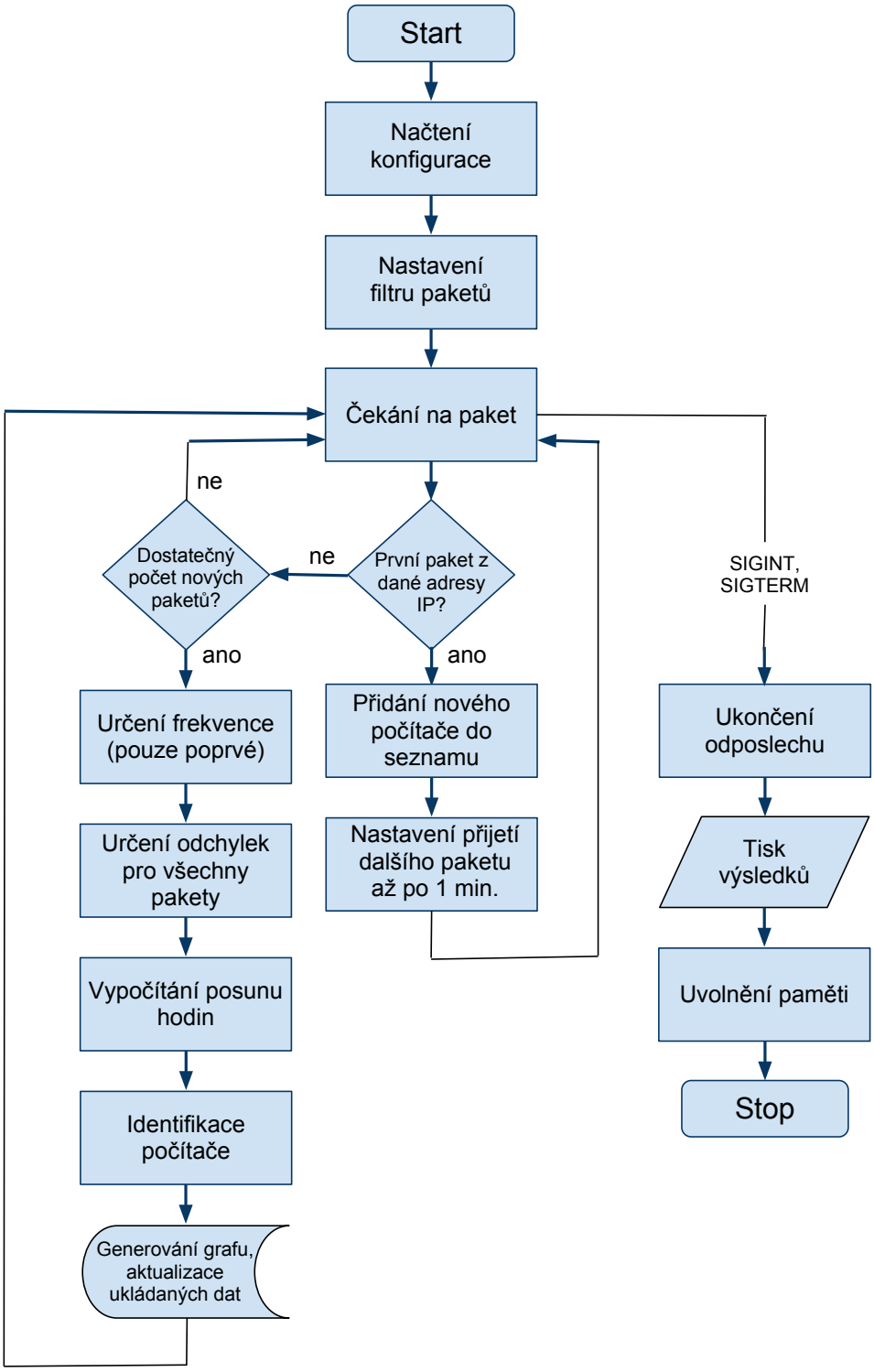
```
'tcp && tcp[12] >= 120'
```

Část `tcp[12]` odkazuje na 12 bajt hlavičky TCP, který obsahuje informaci, kde začínají data paketu a tedy kde končí hlavička TCP včetně pole *TCP Options* (viz sekce 2.3). Minimální velikost hlavičky je 20 bajtů, pokud jsou obsažena i časová razítka, tak 30 bajtů a to nám dává hodnotu tohoto pole (v desítkové soustavě) alespoň 120. Tímto však možnosti filtrování nekončí, uživatel si může pomocí přepínače či konfiguračního souboru nástroje nastavit prakticky libovolný filtr z knihovny *tcpdump* (pro popis filtrů viz manuálové stránky programu *tcpdump*).

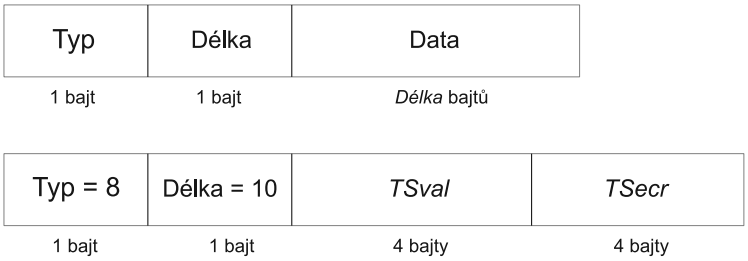
Z přijatých dat potřebujeme získat dvě hodnoty — čas přijetí paketu a časové razítko. Čas přijetí paketu je součástí hlavičky *pcap*, která je přiřazena každému přijatému paketu. Knihovna *libpcap* však neposkytuje rozhraní pro přímý přístup k časovému razítku v paketu TCP, je tedy potřeba každý paket projít a časové razítko vyčíst. Pakety bez časových razítek TCP jsou zahozeny.

Sekce *TCP Options* je součástí hlavičky paketu TCP a může nést doplňující informace. Aby bylo možné *TCP Options* procházet a data vyčíst, mají položky jednotnou formu — viz obrázek 4.2. První bajt vždy určuje typ položky, za ním následuje druhý bajt určující velikost celé položky (včetně polí *Typ* a *Délka*). Pro naše účely je důležitá položka typu 8, obsahující časová razítka (obrázek 4.2). Protože má pole *TCP Options* proměnnou délku (narozdíl od hlavičky TCP, jejíž velikost je vždy 20 bajtů), poslední položka je typu 0, označující konec seznamu [7].

[1] WWW stránky: TCPDUMP/LIBPCAP public repository, `http://www.tcpdump.org/`. Licence: New BSD License.




Obrázek 4.1: Schéma programu `pcf`.




Obrázek 4.2: Obecný formát položky pole *TCP Options* a položka s časovými razítky TCP.

Časová razítka spolu s časem přijetí paketu jsou ukládána do paměti pomocí obousměrně vázaného seznamu. Je samozřejmě možné v jednom okamžiku sledovat více počítačů, v takovém případě existuje jeden seznam pro každý sledovaný počítač. Každý seznam je uvozen hlavičkou, obsahující tyto informace o sledovaném zařízení:

- síťová adresa,
- počet přijatých paketů,
- název (pokud došlo k rozpoznání počítače),
- vypočítaná frekvence generování časových razítek,
- získaný posun vnitřních hodin,
- ukazatele na první a poslední paket seznamu přijatých paketů a
- ukazatel na následující hlavičku (seznam paketů).

Díky použití dynamicky generovaných seznamů není omezen ani počet přijatých paketů, ani počet současně sledovaných počítačů (resp. jediným omezením zůstává množství volné paměti). Současně je pravidelně kontrolována aktivita všech sledovaných zařízení a při dlouhodobé nečinnosti jsou neaktivní seznamy z paměti odstraněny. Stejně tak je možné omezit počet uložených paketů pro jednotlivé počítače, v takovém případě jsou po blocích uvolňovány pakety přijaté před nejdelší dobou. Spolu s provedenými optimalizacemi (viz sekci 4.5) je tak zajištěn bezproblémový chod nástroje po neomezeně dlouhou dobu.

## 4.4 Určení frekvence a odchylek

Jak bylo vysvětleno v podkapitole 3.2, frekvenci získáme jako derivaci časových razítek podle času. K určení derivace tedy nejprve potřebujeme získat určitou množinu hodnot. Experimentálně bylo zjištěno, že 50 hodnot (přijatých časových razítek) je dostatečných pro přesné určení frekvence. Velké nepřesnosti však vznikaly, pokud bylo přijato větší množství paketů okamžitě po započetí sledování. Důvodem byla velmi malá (a nepřesná) relativní hodnota určující dobu měření. Po přijetí prvního paketu tak nástroj začne další pakety zaznamenávat až po jedné minutě.

K určení jednotlivých odchylek, stejně tak jako k výpočtu posunu hodin sledovaného počítače, nedochází po přijetí každého paketu. Především u výpočtu posunu hodin se může jednat o časově poměrně náročnou operaci, výpočty jsou tak prováděny až po získání předem daného počtu paketů (hodnota nastavena pomocí konfiguračního souboru). Odchylky jsou počítány pro každý paket a ukládány jsou nejen do paměti, ale také do souboru pro možnost následného generování grafu.

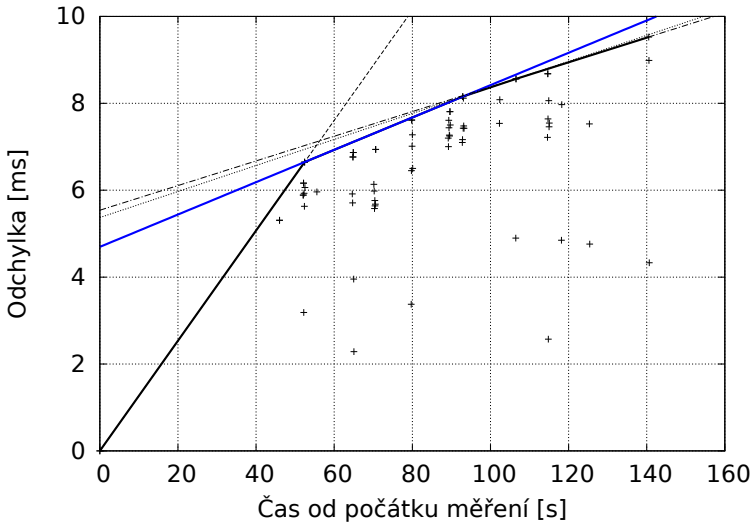
## 4.5 Výpočet posunu hodin

Pro určení posunu hodin sledovaného počítače (a tedy hodnoty identifikující počítač) je potřeba získat rovnici přímky, která shora co nejtěsněji ohraničuje hodnoty odchylek. Odchylka pro každý paket, stejně jako frekvence zvyšování časových razítek, jsou počítány přesně tak, jak je popsáno v podkapitole 3.1. Pro nalezení všech přímek, které shora ohraničují odchylky, je využito principu nalezení konvexní obálky. Z těchto přímek je vybrána ta, která splňuje nejmenší vzdálenost od všech bodů. Úhel, který přímka svírá s osou $x$, je hledaný posun.

Současně s výpočtem posunu hodin je automaticky generován graf, který umožňuje sledovat vývoj posunu hodin s časem v grafické podobě. Ke generování grafů je použit nástroj Gnuplot[2].

### 4.5.1 Nalezení všech shora ohraničujících přímek

Přímek, které shora ohraničují všechny hodnoty odchylek, je nekonečně mnoho. My ale hledáme přímku, která tyto body co nejtěsněji ohraničuje. Množinu takových přímek můžeme získat nalezením konvexní obálky, přičemž pro naše účely nám stačí nalézt horní obal (obrázek 4.3).




Obrázek 4.3: Nalezení všech shora ohraničujících přímek. Přerušovanými čárami jsou znázorněny jednotlivé přímky, tučně je zvýrazněna konvexní obálka (horní obal). Modrou barvou je zobrazena výsledná přímka, určující posun hodin.

K nalezení horního obalu je využit *Graham scan* algoritmus [6]. Z množiny bodů v rovině hledáme tzv. extrémní body, určující vrcholy mnohoúhelníku, kde všechny ostatní body leží uvnitř tohoto mnohoúhelníku. Prvním z těchto bodů je v našem případě bod se souřadnicemi [0,0], který je nejlevějším bodem celé množiny a zřejmě tak bude součástí obálky. Pro

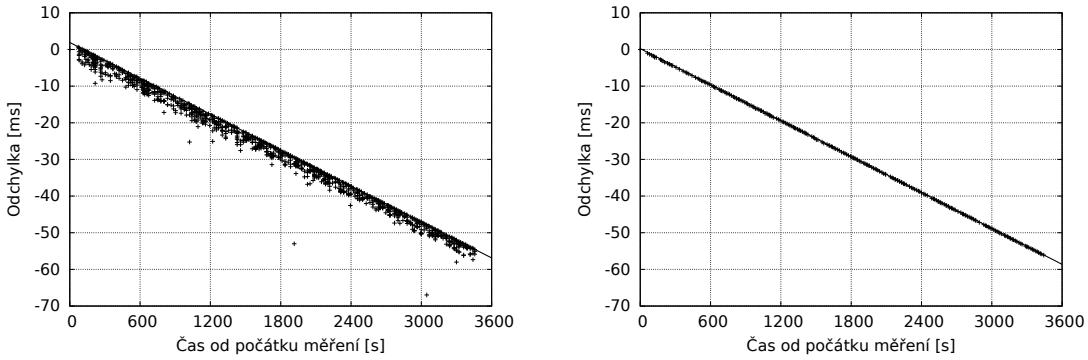
[2]WWW stránky: Gnuplot homepage, `http://www.gnuplot.info/`. Licence: Gnuplot's copyright.

nalezení dalších bodů je využito právě *Graham scan* algoritmu, konkrétně testování kritéria levotočivosti (v našem případě pravotočivosti) [1]:

- Pokud uspořádaná trojice bodů $p_{j-1}$, $p_j$, $p_{j+1}$ splňuje kritérium pravotočivosti, pak body zřejmě tvoří (ale nemusí) konvexní obálku. Následuje testování trojice $p_j$, $p_{j+1}$, $p_{j+2}$.
- Pokud pro danou trojici bodů neplatí kritérium pravotočivosti, bod $p_j$ konvexní obálku určitě netvoří a může být odstraněn. Testování pokračuje s body $p_{j-2}$, $p_{j-1}$, $p_{j+1}$.

Tímto způsobem postupně projdeme přes všechny body a získáme množinu extrémních bodů. Rovnice přímek dostaneme vždy ze dvou sousedících bodů. Časová složitost tohoto algoritmu je $O(n \cdot log(n))$, je tedy vhodný i pro rozsáhlé množiny bodů [6].

Pro urychlení výpočtu a snížení paměťových nároků programu je před samotným výpočtem zredukována posloupnost odchylek o hodnoty, které jsou pro nalezení horního obalu evidentně nepotřebné[3]. Při rostoucím posunu jsou tak odstraněny pakety, jejichž odchylka má menší hodnotu, než jakou má paket přijatý dříve v čase. Stejně tak pro klesající posloupnost, zde je však potřeba postupovat od posledního paketu k prvnímu (obrázek 4.4).




(a) Bez redukce: ≈1150 paketů

(b) S redukcí: ≈240 paketů

Obrázek 4.4: Redukce paketů.

### 4.5.2 Určení výsledného posunu

Výsledný posun udává sklon přímky, jejíž vzdálenost od všech bodů (odchylek) je nejmenší. Základem je tak sečtení vertikálních vzdáleností všech bodů od jednotlivých přímek a vybrání té s nejmenší výslednou hodnotou.

I zde bylo provedeno několik optimalizací. Pokud při výpočtu součtu vzdáleností dojde k překročení dosavadní minimální hodnoty, je výpočet předčasně ukončen — zřejmě se nebude jednat o minimální hodnotu. Experimenty bylo navíc zjištěno, že první a poslední z ohraničujících přímek s velkou pravděpodobností výsledné nebudou a naopak mívají součet značně vyšší. Jako první je tak výpočet proveden pro přímku uprostřed ohraničení.

[3] Jedná se o pakety více či méně zpožděné, které jsou pro určení posunu hodin sledovaného počítače nepoužitelné.

S tímto souvisí i další optimalizace — pokud je sklon přímky příliš strmý, je daná přímka automaticky přeskočena (jedná se o důležitou podmínku, snadno by tak totiž mohlo dojít k přetečení čítače).

## 4.6 Uložení a rozpoznání počítače

Pro uložení počítačů byl zvolen formát XML, k práci s tímto formátem nástroj využívá otevřenou knihovnu *Libxml2*[4]. Formáty souborů sledovaných (aktivních) a uložených počítačů jsou uvedeny v příloze B.

Obsah souboru se sledovanými počítači se dynamicky mění podle aktivní síťové komunikace (po jaké době bez aktivní komunikace je sledovaný počítač odebrán lze nastavit v konfiguračním souboru). Kdykoliv je pak možné kterýkoliv počítač permanentně uložit do databáze a pojmenovat. Rozpoznávání počítačů probíhá mezi právě sledovanými počítači a počítači uloženými v databázi, přičemž identifikace v rámci právě sledovaných počítačů má přednost před procházením uložených počítačů. Pokud tak dojde k rozpoznání dvou aktivních počítačů, jsou identifikovány navzájem.

Posun vnitřních hodin sledovaného počítače je vyjádřen jedním reálným číslem (podkapitola 3.2). Protože se jedná o měření velice drobných odchylek (v řádu desítek milisekund za hodinu), hodnota nemusí vždy vyjít úplně přesně — jak bude také ukázáno v kapitole 5. V implementaci nástroje je pro uložení výsledného posunu použit typ *double* s přesností na šest desetinných míst a na rozpoznání počítače má velký vliv hodnota zvoleného prahu. Pokud dojde k rozpoznání více počítačů najednou, je vždy vybrán ten s nejbližší hodnotou posunu hodin.

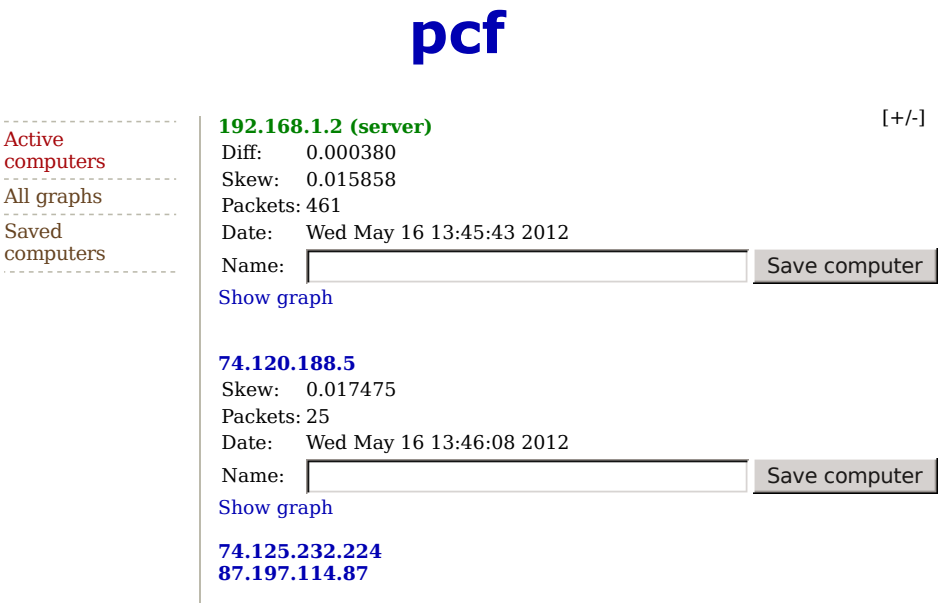
## 4.7 Grafické uživatelské rozhraní

Pro snažší zobrazení a zpracování výsledků bylo k nástroji `pcf` vytvořeno grafické uživatelské rozhraní ve formě webových stránek (HTML, PHP, JavaScript, CSS). Na obrázku 4.5 je vidět hlavní stránka, která zobrazuje právě sledované počítače. Zelenou barvou jsou zvýrazněny rozpoznané počítače, v závorce za adresou IP je navíc uveden název počítače, s kterým byl daný počítač identifikován. Za seznamem rozpoznaných počítačů následuje seznam ostatních aktivních počítačů (adresy zobrazeny modrou barvou). Při kliknutí na adresu IP daného počítače je možné zobrazit podrobnosti (naměřený posun hodin, počet přijatých, resp. použitých paketů, datum a čas naposledy přijatého paketu), graf průběhu měření posunu hodin počítače a je zde také možnost pro uložení počítače do databáze.

Sekce *All graphs* nabízí zobrazení všech grafů právě sledovaných počítačů najednou. Detaily o počítačích jsou uvedeny v hlavičce každého z grafů. Protože jsou grafy, které `pcf` automaticky generuje, ve formátu *PostScript*, nástroj pravidelně volá připravený Bash skript `gen_pics.sh`, který pomocí programu Ghostscript[5] konvertuje grafy na obrázky ve formátu *jpeg*.

Uživatelské rozhraní umožňuje také spravovat databázi uložených počítačů, k čemuž slouží sekce *Saved computers*.

[4] WWW stránky: The XML C parser and toolkit of Gnome, `http://xmlsoft.org/`. Licence: MIT License.
[5] WWW stránky: Ghostscript: Ghostscript, `http://www.ghostscript.com/`. Licence: GPL3.




Obrázek 4.5: Webové uživatelské rozhraní.

# Kapitola 5

# Výsledky experimentů

Základem experimentů bylo získání databáze počítačů s posunem jejich vnitřních hodin. K tomuto účelu byl nasazen server, kde mohl implementovaný nástroj běžet neustále a byl dostupný odkudkoliv. Pro zajištění síťového provozu byl vytvořen jednoduchý skript v jazyce Python, simulující reálnou komunikaci mezi sledovaným a sledujícím počítačem. Experimenty byly provedeny s počítači různých značek, konfigurací, operačních systémů, umístění atd.

Při provádění experimentů vyšly najevo dva problémy. Prvním z nich bylo nepoužívání doporučení z RFC 1323 některými operačními sytémy. Jednalo se především o počítače s operačním systémem Windows, které ve standardní konfiguraci časová razítka do hlaviček paketů TCP neuváděla. Druhým problémem byla časová synchronizace. Ačkoliv podle [8, 21] by neměla mít synchronizace času protokolem NTP vliv na generování časových razítek TCP, provedené experimenty s aktuálními operačními systémy ukázaly, že tomu tak být nemusí.

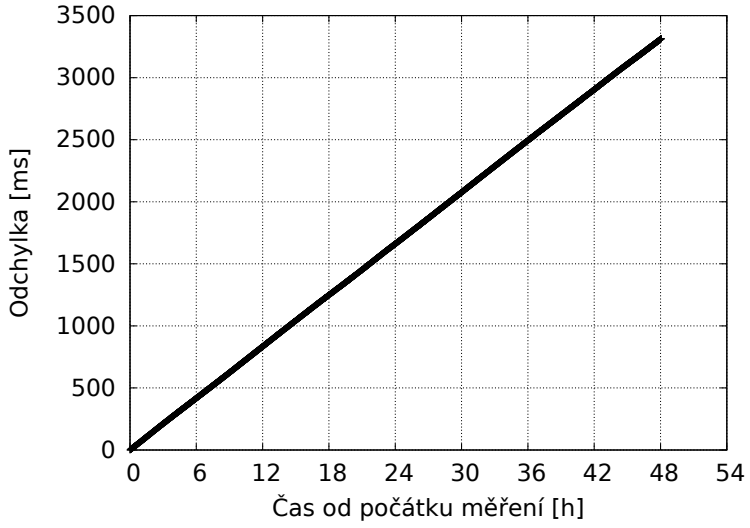
## 5.1 Doba měření

Ve většině případů stačilo pro přesné určení posunu vnitřních hodin sledovaného počítače měření po dobu jedné hodiny, poté se již hodnota měnila minimálně. Délka měření se navíc ukázala jako důležitější měřítko pro určení přesnosti posunu než samotný počet přijatých paketů. To v praxi znamená, že pro identifikaci počítače nám stačí přijmout několik paketů v jednom okamžiku a několik paketů například za hodinu a z těchto pár paketů jsme schopni změřit posun jeho vnitřních hodin. Pokud měření trvá příliš krátkou dobu (v řádu minut), větší množství paketů nám k přesnějšímu určení hodnoty nijak nepomůže.

Předpokládejme nyní, že máme dostatečný síťový provoz. Pokud nám jde o verifikaci jednoho konkrétního počítače, již po dvou minutách sledování jsme schopni určit, zda se může jednat o daný počítač, nebo se o daný počítač určitě nejedná. Pro přesnější určení výsledného posunu je však potřeba měření v řádu desítek minut. Hodnoty průběhu měření posunu vnitřních hodin jednoho sledovaného počítače jsou uvedeny v tabulce 5.1, příslušné grafy jsou součástí přílohy C. Na obrázku 5.1 je zobrazen průběh měření počítače po dobu 48 hodin.

Tabulka 5.1: Průběžné výsledky naměřeného posunu hodin v čase.

| **Uplynulý čas [min]** | **Posun hodin [ppm]** | **Rozdíl proti 60 min. [ppm]** |
|---|---|---|
| 2 | 15,932 | -0,766 |
| 3 | 16,128 | -0,570 |
| 5 | 16,801 | 0,103 |
| 10 | 16,465 | -0,233 |
| 15 | 16,521 | -0,177 |
| 30 | 16,698 | 0,000 |
| 60 | 16,698 | — |
| 120 | 16,676 | -0,022 |




Obrázek 5.1: Průběh měření počítače s konstantním posunem vnitřních hodin po dobu 48 hodin.

## 5.2 Sledující počítač

Implementovaný nástroj běžel na počítači s operačním systémem Debian Linux 5.0.9 s verzí jádra 2.6.26. Systém používal synchronizaci času pomocí démona běžícího nad protokolem NTP, hodnoty naměřených posunů hodin sledovaných počítačů jsou tak vzhledem k reálnému času.

## 5.3 Sledované počítače

V této podkapitole si uvedeme výsledky provedených experimentů. Kromě testů s počítači stejné konfigurace (sekce 5.3.4) a testování vlivu zatížení sítě (sekce 5.3.8) byly experimenty prováděny v reálném prostředí za běžného síťového provozu.

Jak již bylo zmíněno výše, operační systémy Windows implicitně časová razítka do paketů TCP neuvádějí. Povolení/zakázání časových razítek v hlavičkách paketů TCP se provádí úpravou registrů, konkrétně je potřeba přidat řetězec *Tcp1323Opts* s hodnotou „2“ (případně „3“) do klíče registru:

```
HKEY_LOCAL_MACHINE\System\CurrentControlSet\Services\Tcpip\Parameters
```

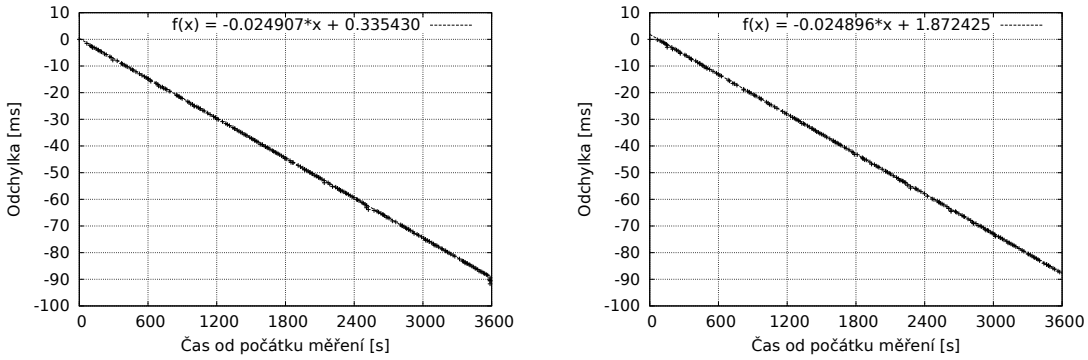
U systému Linux lze dané nastavení měnit za běhu pomocí změny parametru jádra:

```
echo 1 > /proc/sys/net/ipv4/tcp_timestamps
```

Všechny testované počítače s operačním systémem Linux, BSD nebo OpenSolaris měly implicitně tuto hodnotu nastavenu na „1“ a tedy aktivní.

### 5.3.1 Ukázkový případ

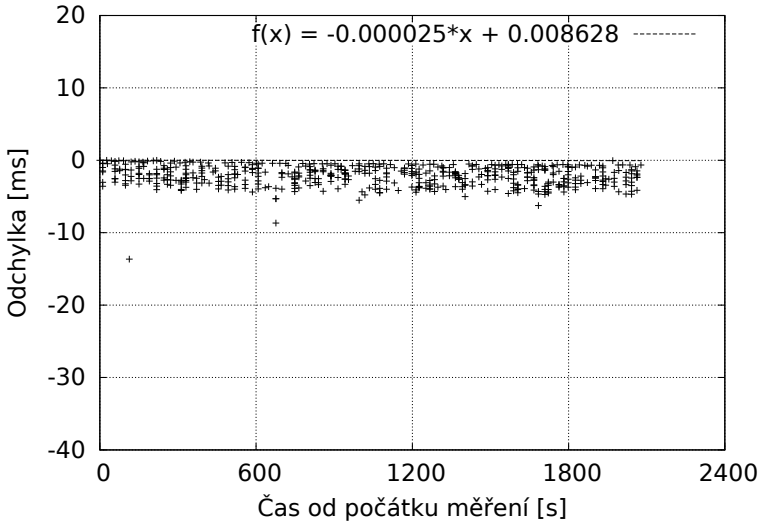
Nejlepších výsledků dosahovala identifikace počítačů s operačním systémem Linux bez zapnuté synchronizace času. Na obrázku 5.2 jsou zobrazeny grafy dvou měření jednoho počítače v rámci lokální bezdrátové sítě. První hodinové měření sloužilo pro naučení počítače a hned na to byla provedena identifikace. Po hodině byl rozdíl obou naměřených posunů hodin 0,011 ppm.




Obrázek 5.2: Naučení a identifikace počítače, Ubuntu Linux (bez NTP). Rozdíl posunu hodin po hodinovém měření je 0,011 ppm.

### 5.3.2 Počítač se synchronizovanými hodinami

Vliv synchronizace času pomocí protokolu NTP na hodnotu časových razítek TCP je u různých operačních systémů rozdílný. Až samotné experimenty ukázaly, že počítače s operačním systémem Linux, používající časovou synchronizaci démonem NTP (po dostatečně dlouhou dobu), nevykazují prakticky žádný posun vnitřních hodin (obrázek 5.3). Naopak u operačních systémů FreeBSD (testováno s verzemi 8.2 a 8.3) nemá synchronizace času na získaný posun vnitřních hodin žádný vliv. U systémů MS Windows sice k synchronizaci času dochází jednorázově, přesto však dochází k větším či menším úprávám posunu vnitřních hodin. Z tohoto zjištění vyplývá fakt, že kromě FreeBSD nynější operační systémy generují časová razítka TCP až po časové synchronizaci. V takovém případě je nemožné sledovaný počítač na základě časových razítek TCP identifikovat.




Obrázek 5.3: Počítač s operačním systémem Linux a se synchronizací času pomocí démona NTP.

Na obrázku 5.4 je zobrazen vývoj měření posunu vnitřních hodin vzdáleného počítače s operačním systémem Arch Linux. Nejprve počítač nepoužíval žádnou synchronizaci času a vykazoval konstantní posun hodin. Po půl hodině došlo ke spuštění démona NTP a z grafu je jasně patrné, jak v průběhu několika hodin docházelo k postupné korekci posunu hodin, která se projevila i na hodnotách časových razítek získaných z hlaviček paketů TCP. Po čtyřech hodinách byl démon NTP opět zastaven a přestalo tak docházet k další korekci času, vnitřní hodiny sledovaného počítače však v dalším průběhu měření vykazovaly naposledy „zkorigovaný“ posun. Až po restartu systému byl naměřený posun hodin znovu stejný jako na začátku měření a tedy bez jakékoliv korekce.

### 5.3.3 Jednorázová synchronizace pomocí `ntpdate`

Synchronizace hodin u systémů Linux nemusí fungovat vždy jako démon, snažící se průběžně kompenzovat posun vnitřních hodin počítače vzhledem k reálnému času. Druhou možností

je pouze synchronizovat aktuální čas v určitých časových okamžicích — jednou za den, za týden, při startu systému apod. Obrázek 5.5 ilustruje, jak se synchronizace času v hodinových intervalech projeví na posunu hodin, resp. časových razítkách TCP. Konkrétně se jedná o počítač s operačním systémem CentOS Linux a o synchronizaci času se stará program `ntpdate`, spouštěný každou hodinu démonem `cron`. Korekce navíc nenastane nárazově v jednom okamžiku, ale dochází k postupnému upravení hodnoty (v ukázkovém případě trvá korekce asi 4 minuty). Protože při tomto typu synchronizace zůstává posun vnitřních hodin počítače po většinu času nezměněný, je možné ho identifikovat.

### 5.3.4 Vliv operačního systému a počítače se stejnou konfigurací

Následující experimenty byly provedeny v počítačové laboratoři v rámci lokální sítě. Všechny sledované počítače měly naprosto stejnou konfiguraci (jak hardwarovou, tak softwarovou), byly umístěny ve stejné místnosti a měření všech počítačů probíhalo současně. V prvním testu běžel na počítačích operační systém FreeBSD 8.2, ve druhém Windows XP Professional Service Pack 3 a ve třetím Debian Linux (verze jádra 2.6.32). Všechna měření trvala jednu hodinu. U systémů FreeBSD a Linux nebyl čas synchronizován žádným způsobem, ve Windows pak pomocí NTP jednou týdně. Provedená měření ukázala (spolu s výsledky ze sekce 5.3.6), že je hodnota naměřeného posunu vnitřních hodin počítače závislá na běžícím operačním systému.

#### FreeBSD 8.2

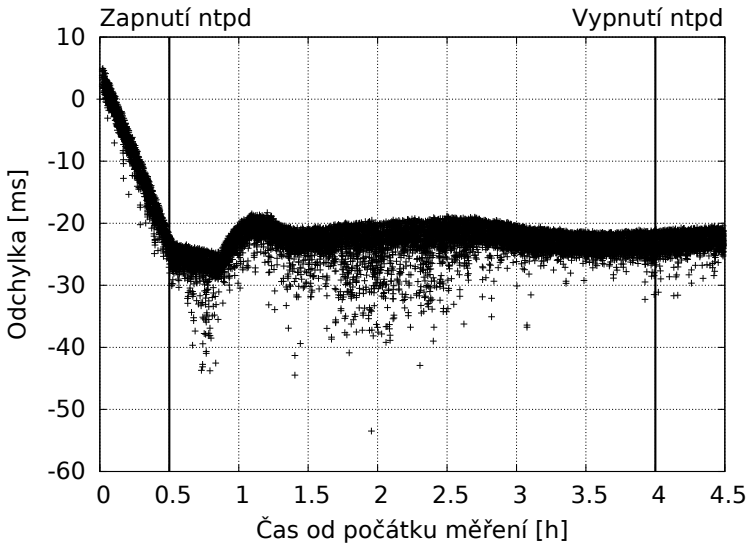
Jak je vidět v tabulce 5.2, kromě jednoho počítače (pc3) jsou naměřené posuny hodin velmi podobné. Rozdíl mezi pc1 a pc2 je dokonce pouze 0.005 ppm, počítače mimo pc3 jsou tak v praxi nerozlišitelné. Všechny posuny byly konstantní po celou dobu měření a počítače vykazovaly prakticky stejný posun vnitřních hodin i při opakovaném měření po uplynutí několika týdnů.

Tabulka 5.2: Naměřené posuny hodin pěti počítačů se stejnou konfigurací — operační systém FreeBSD 8.2.

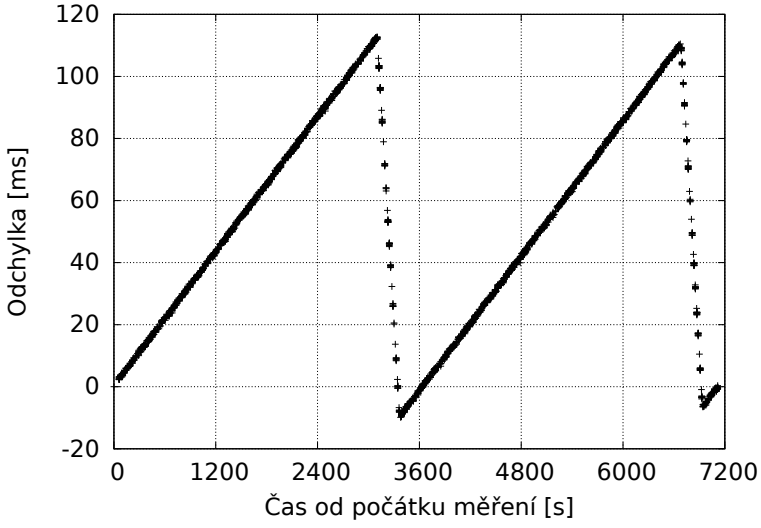
| **Počítač** | **Posun hodin (ppm)** |
|---|---|
| pc1 | -277,313 |
| pc2 | -277,308 |
| pc3 | -270,529 |
| pc4 | -277,789 |
| pc5 | -277,846 |

#### Windows XP Professional SP3

Tabulka 5.3 ukazuje výsledky měření u operačního systému Windows XP. Naměřené posuny jsou naprosto odlišné než u operačního systému FreeBSD a také rozdíly mezi jednotlivými stanicemi jsou značně větší. I v tomto případě byly po dobu měření všechny posuny konstantní a to i po restartu systému. Při opakování měření po několika týdnech byl však u jednotlivých stanic naměřen velký rozdíl posunu hodin (až 7 ppm) oproti původnímu měření, což činí daný počítač neidentifikovatelný.




Obrázek 5.4: Postupná korekce posunu hodin počítače pomocí démona NTP.




Obrázek 5.5: Synchronizace času v hodinových intervalech.

Tabulka 5.3: Naměřené posuny hodin pěti počítačů se stejnou konfigurací — operační systém Windows XP Professional SP3.

| Počítač | Posun hodin (ppm) |
|---|---|
| pc1 | -18,408 |
| pc2 | -37,285 |
| pc3 | -22,766 |
| pc4 | -22,256 |
| pc5 | -26,640 |

#### Debian Linux

Debian Linux nainstalovaný na stanicích v laboratoři vykazoval naprosto rozdílný posun při každém startu systému. Tabulka 5.4 je tak uvedena spíše pro úplnost, protože fakt, že mají všechny počítače stejnou konfiguraci, je při daném chování systému bezvýznamné.

Tabulka 5.4: Naměřené posuny hodin pěti počítačů se stejnou konfigurací — operační systém Debian Linux.

| Počítač | Posun hodin (ppm) |
|---|---|
| pc1 | 59,450 |
| pc2 | 31,938 |
| pc3 | -67,035 |
| pc4 | -90,805 |
| pc5 | -184,733 |

### 5.3.5 Virtuální versus reálný systém

Experimenty byly provedeny také s virtualizovanými operačními systémy. Jako virtualizační nástroj byl použit Oracle VM VirtualBox 4.1.8 (Open Source Edition)[1] s následujícími virtualizovanými systémy:

- Tiny Core Linux (3x), verze jádra 3.0.3,
- CentOS 6.0 (2x), verze jádra 2.6.32,
- FreeBSD 8.2,
- BackTrack 4 R2 (Ubuntu Linux), verze jádra 2.6.35.

Hostitelským operačním systémem byl Arch Linux s verzí jádra 3.2.8.

V první řadě je potřeba rozlišit, jakým způsobem bude virtualizovaný systém přistupovat k síťovému rozhraní. Dva základní režimy jsou *NAT* a *síťový most*. Při použití režimu *NAT* virtualizační prostředí simuluje práci směrovače včetně DHCP serveru a vytváří tak rozhraní mezi hostitelským a virtualizovaným systémem. Virtualizovaný systém se připojuje k tomuto virtuálnímu směrovači, od DHCP serveru obdrží vlastní síťovou adresu (obvykle z jiného adresového prostoru než jaký používá hostitelský operační systém) a jakákoliv síťová komunikace prochází přes tento směrovač. U *síťového mostu* má virtualizovaný systém přímý přístup k ovladači síťové karty a se sítí tak pracuje sám, bez zásahu hostitelského systému [3].

[1] WWW stránky: Oracle VM VirtualBox, `http://www.virtualbox.org/`. Licence: GPL.

#### Režim *NAT*

Při snaze identifikovat virtualizovaný systém připojený k síti v režimu *NAT* dojde k identifikaci hostitelského počítače, nehledě na běžícím virtualizovaném systému. O samotné posílání paketů včetně generování časových razítek TCP se totiž stará hostující operační systém.

Pokud se pokusíme na takto nastaveném virtualizovaném systému spustit nástroj pro identifikaci, narazíme hned na dva problémy. Prvním z nich je nemožnost přesměrovat porty menší než 1024 (viz [3]). Druhou komplikací je fakt, že virtuální směrovač vytvořený virtualizačním nástrojem (v našem případě VirtualBox) odstraňuje z hlaviček všech příchozích paketů TCP pole *TCP Options*, kde se nacházejí časová razítka.

Z těchto důvodů není možné virtualizovaný systém v síťovém režimu *NAT* ani identifikovat, ani využít jako sledující počítač.

#### Režim *Síťový most*

Jiná situace nastává, pokud virtualizovaný systém přistupuje k síti v režimu *síťový most*. Pro spuštění identifikačního nástroje není potřeba nic nastavovat, virtualizační nástroj umožňuje přímý přístup k síťovému rozhraní a vše se tak chová stejně jako při práci s rozhraním bez virtualizace.

Při identifikaci však i přes přímý přístup k síťovému rozhraní vykazovaly jednotlivé systémy naprosto odlišné posuny vnitřních hodin a to i v případě, že se jednalo o dva totožné (naklonované) systémy. Díky tomu byl každý ze systémů identifikován jako jiný fyzický počítač. Pokud byl navíc virtualizovaný systém restartován, po novém startu byla hodnota posunu jeho vnitřních hodin odlišná. Jedinou výjimkou zde byl systém FreeBSD, který vykazoval stále stejný posun.

### 5.3.6 Operační systém spouštěný z optického média

Dalším z provedených experimentů byla identifikace systému běžícího z optického média, tzv. *live* operačního systému. Na sledovaném počítači byly nainstalovány operační systémy Arch Linux s jádrem 3.2.8 a Windows 7, přičemž stejně jako v případě počítačů v laboratoři (popsané v sekci 5.3.4) byly naměřené posuny hodin u obou systémů odlišné.

Jako *live* operační systémy byly zvoleny FreeBSD 8.3, Fedora 16 (verze jádra 3.1.0) a PCLinuxOS s jádrem 2.6.38.

Měření se systémem FreeBSD potvrdilo výsledky z laboratoře a i FreeBSD spouštěné z optického média vykazovalo odlišný posun než nainstalované sytémy Linux i Windows.

Z optického média spuštěné systémy Fedora 16 a PCLinuxOS naopak ukázaly, že pokud se jedná o stejný typ operačního systému na jednom počítači, je vykazovaný posun hodin ve všech případech stejný. Rozdíl naměřených posunů mezi nainstalovaným Arch Linuxem a systémem Fedora 16 byl 0,038 ppm a 0,457 ppm oproti systému PCLinuxOS. Roli nehraje ani různá frekvence generování časových razítek TCP u jednotlivých verzí systému, například u systému Fedora byla tato frekvence 1000 Hz a u systému Arch Linux 300 Hz.

### 5.3.7 Chytré mobilní zařízení

Mobilní telefon s operačním systémem Android 2.3.7 vykazoval nejmenší posun hodin ze všech měřených zařízení: -9,56 ppm (necelých 35 milisekund za hodinu). Na druhou stranu experimenty ukázaly, že i tato zařízení mají měřitelný konstantní posun vnitřních hodin a

je tak možné je pomocí časových razítek TCP identifikovat. Rozdíl dvou nezávislých měření tohoto telefonu byl 0,779 ppm.

Druhý testovaný přístroj (Android 2.3.5) měl zapnutou automatickou synchronizaci času od operátora, i přesto však vykazoval konstantní posun hodin. Tento typ synchronizace tak neměl na identifikaci chytrého mobilního přístroje žadný vliv a oba přístroje bylo možné rozpoznat.

### 5.3.8 Další provedené experimenty

Několik dalších experimentů bylo provedeno s přenosným počítačem (notebookem) — vliv typu napájení, připojení, umístění a zatížení počítače na hodnotu posunu vnitřních hodin. Použitými operačními systémy byly Arch Linux (jádro 3.2.8) a Windows 7.

#### Typ napájení

Výsledky měření ukázaly, že na hodnotě posunu hodin se ani u jednoho z operačních systémů nijak neprojeví, zda je počítač napájen ze sítě či z baterie. Napájení ze sítě bylo odebráno jak v průběhu měření, tak i před samotným startem počítače. Ani v jednom případě však výsledná hodnota nebyla nijak ovlivněna a posun hodin při běhu počítače z baterie byl stejný jako při napájení ze sítě.

#### Typ připojení

Sledovaný počítač měl dvě síťová rozhraní, drátové a bezdrátové. Obě tato rozhraní udávala stejný posun vnitřních hodin.

#### Umístění (vzdálenost)

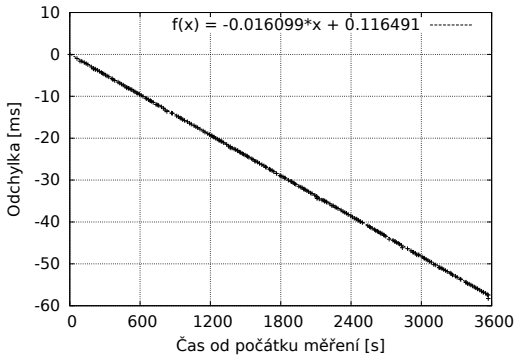
Ani změna umístění počítače neměla velký vliv na naměřený posun jeho vnitřních hodin. Rozdíl mezi naměřenými posuny při měření v rámci lokální sítě oproti měření přes internet byl 0,152 ppm (vzdálenost 9 hopů) a 0,545 ppm při vzdálenosti 11 hopů. Kvůli většímu množství zpožděných (a pro měření nepoužitelných) paketů je však potřeba měření po delší dobu, abychom získali dostatečný počet paketů s minimálním zpožděním v rámci daného spojení.

#### VPN

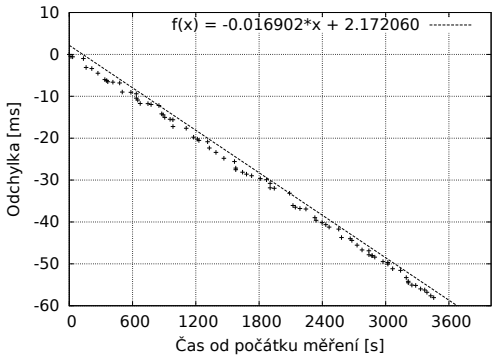
Dalším provedeným experimentem byla identifikace počítače připojeného ke vzdálené síti pomocí protokolu VPN a přistupujícím do sítě internet přes vzdálenou bránu. Ta byla navíc od sledujícího počítače vzdálena 12 hopů. Z grafu průběhu měření bylo vidět poměrně velké kolísání zpoždění paketů a ještě po půl hodině měření docházelo k větším změnám výsledného posunu (viz obrázek 5.6). I rozdíl naměřeného posunu vzhledem k posunu určenému v rámci lokální sítě byl větší — 0.803 ppm

#### Počítače za překladačem adres (NAT)

Jedním z dalších využití rozdílného posunu vnitřních hodin počítačů může být rozlišení počítačů za překladačem adres (NAT). Pro zajištění této funkcionality by ovšem muselo dojít k zásadnější změně identifikačního nástroje. Pokud však počítače nebyly spuštěny současně, může dojít k jejich rozlišení pouze na základě různých rozsahů aktuálních hodnot




(a) Měření počítače v rámci lokální sítě




(b) Počítač přistupující do sítě internet přes VPN bránu

Obrázek 5.6: Vliv větší vzdálenosti a VPN připojení na výsledný posun hodin.

časových razítek TCP. Příklad je ukázán na obrázku 5.7, kdy oba počítače sice vykazují velice podobný posun vnitřních hodin, spuštěny však byly s časovým odstupem čtyř sekund.

### Vliv zatížení sítě

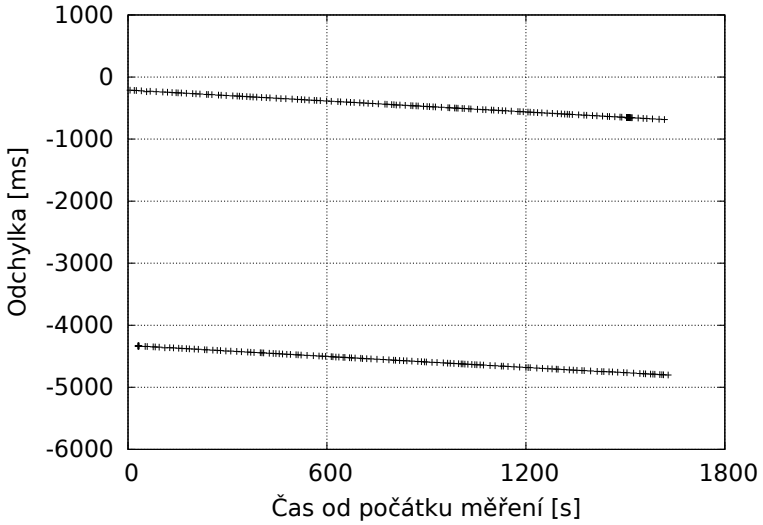
Při běžném provozu nemá zatížení sítě na získání posunu vnitřních hodin sledovaného počítače prakticky žádný vliv. Jedná se ovšem o případ, kdy je zatížení konstantní. Experimenty ukázaly, že pokud dojde k nárazovému zvýšení zatížení sítě, projeví se to i na měření posunu vnitřních hodin počítače. Obrázek 5.8 ukazuje část měření, kdy po uplynutí jedné minuty došlo k nárazovému zvýšení zatížení sítě (pomocí hardwarového generátoru paketů). Po uplynutí zhruba další minuty je v grafu vidět postupné odchylování od původního konstantního posunu, což je následkem zvětšování zpoždění jednotlivých paketů. Po šesti minutách došlo k přetížení a zhroucení celé sítě. Komunikace se znovu, tentokrát již bez přidaného zatížení, rozběhla po deseti minutách, kde již sledovaný počítač opět vykazoval konstantní posun vnitřních hodin.

### Zatížení počítače a teplota

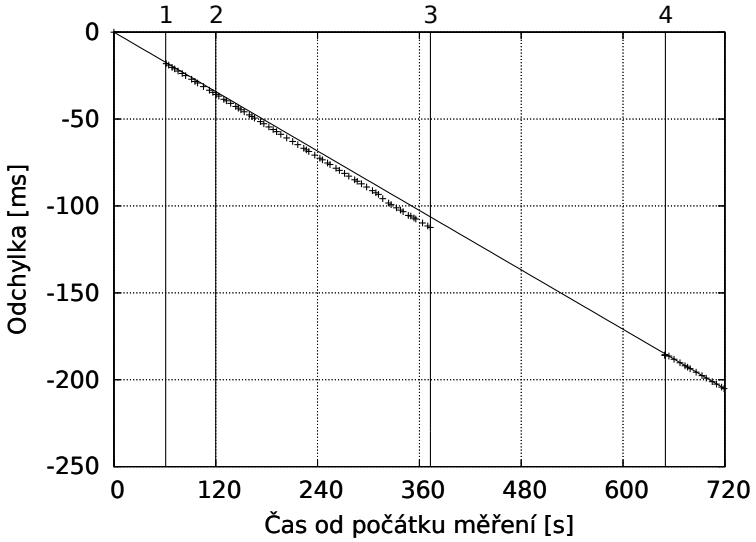
Posun vnitřních hodin sledovaného počítače byl měřen také při různém zatížení tohoto počítače. Spolu se zátěží se měnila i teplota procesoru (a celého zařízení), na hodnotu naměřeného posunu hodin to však prakticky žádný vliv nemělo — jak je patrné z tabulky 5.5, rozdíl ve změně zatížení a teplotě oproti běžnému provozu není ani v jednom případě větší než 0,3 ppm.

## 5.4 Nasazení nástroje v reálném prostředí

Nástroj se podařilo vyzkoušet při běhu v reálném prostředí, když byl nasazen na síťové sondě studentských kolejí. Největším problémem zde byl velmi vysoký datový tok, bylo tedy potřeba správně zvolit a nastavit filtrování, aby bylo co nejvíce paketů odfiltrováno již na nejnižší úrovni. Samotný nástroj, jakožto čistě softwarové řešení, se s příliš velkým




Obrázek 5.7: Dva počítače za překladačem adres (NAT), spuštěné s časovým odstupem 4 sekundy.




Obrázek 5.8: Vývoj měření posunu vnitřních hodin sledovaného počítače při nárazovém přetížení sítě. 1: Spuštění generátoru paketů. 2: Odchylování od původního konstantního posunu. 3: Přetížení a zhroucení sítě. 4: Měření pokračuje s běžným síťovým provozem.

Tabulka 5.5: Rozdíl posunu hodin při různém zatížení a teplotě oproti běžnému provozu. Při nízké zátěži neběžel na sledovaném počítači kromě operačního systému žádný další program, při vysoké zátěži se počítač staral o přehrávání videa s vysokým rozlišením (Full HD). Průměrná frekvence a teplota jsou hodnoty procesoru.

| **Zatížení** | **Prům. frek.** | **Prům. tep.** | **Posun [ppm]** | **Rozdíl [ppm]** |
|---|---|---|---|---|
| Nízké | 1,5 GHz | 38°C | -39,146 | -0,231 |
| Běžný provoz | 1,6 GHz | 45°C | -39,377 | — |
| Vysoké | 2,7 GHz | 75°C | -39,592 | 0,215 |

datovým tokem vyrovnat nedokáže. Používání programu `pcf` na síťových sondách je součástí navazujícího výzkumu.

## 5.5 Ideální práh a počet rozlišitelných počítačů

Důležitou součástí experimentů bylo nalezení ideálního prahu pro rozlišení jednotlivých počítačů. Z výsledků testů vyplývá, že tato hodnota leží mezi 0,8 — 1,2 ppm. Větší rozdíl než 1,2 ppm se v rámci jednoho počítače vyskytoval minimálně a zbytečně by tak nastavení vyššího prahu snižovalo celkovou rozlišovací schopnost. Naopak nižší hodnota než 0,8 ppm může způsobovat nerozpoznání stejného počítače. Několik počítačů bylo průběžně testováno a měřeno i s odstupem v řádu měsíců a docházelo k rozdílům právě až kolem 0,8 ppm.

Pro určení (teoretického) maximálního počtu rozlišitelných počítačů je podstatné, v jakém rozsahu se v praxi vyskytují posuny vnitřních hodin počítačů. Nejvyšší hodnoty (vzhledem k reálnému času) se pohybovaly okolo 500 ppm, nejnižší kolem 9 ppm. S rozlišovacím prahem nastaveným na 1 ppm a faktem, že hodnoty posunů mohou být jak kladné, tak záporné, bychom teoreticky mohli být schopni rozlišit téměř 1000 zařízení. V praxi se však nejvíce posunů hodin pohybovalo v rozmezí 12 — 110 ppm a reálně jsme schopni rozlišit maximálně kolem 100 zařízení. I u takovéto databáze je již značná pravděpodobnost, že bude mít více počítačů velice podobné hodnoty posunů vnitřních hodin.

# Kapitola 6

# Závěr

Tato práce je zaměřena na identifikaci počítačů na základě časových informací získaných ze síťové komunikace. Tato technika byla již dříve zkoumána [8, 15, 21], cílem této práce bylo navázat na tyto výzkumy a zjistit použitelnost dané techniky v praxi.

Jako zdroj časových informací byla zvolena časová razítka TCP. Zpracovány však byly i další možnosti získání časových značek od sledovaného počítače. Důkladně byla prostudována a popsána také technika určení výsledného posunu vnitřních hodin počítače, díky čemuž mohla být do vyvíjeného nástroje implementována celá řada optimalizací.

Výstupem praktické části je program schopný na základě síťového provozu samostatně identifikovat počítače. Nástroj je implementován jako démon, může tedy běžet po neomezeně dlouhou dobu. Veškeré důležité informace o sledovaných počítačích nástroj zaznamenává do souboru ve formátu XML a poskytuje tak rozhraní pro další zpracování. Pro jednodušší sledování a zpracování výsledků identifikace bylo nad démonem vytvořeno webové rozhraní. Pro každý počítač je navíc automaticky generován graf, umožňující sledovat průběh měření posunu jeho vnitřních hodin v grafické podobě.

Důležitou částí práce bylo provedení velkého množství experimentů. Od testování různých operačních systémů, počítačů se zapnutou i vypnutou synchronizací času, virtualizovaných systémů, přes měření chytrých mobilních telefonů, operačních systémů spouštěných z optického média, až po určování vlivu vzdálenosti, typu připojení, zatížení počítače a sítě. Program se podařilo nasadit i v reálném prostředí studentských kolejí.

Z dosažených výsledků vyplývá, že identifikace počítačů na základě časových razítek TCP není v současné době a aktuálními operačními systémy tak ideální, jak je popsáno například v [8]. Mezi hlavní problémy patří rozdílný posun vnitřních hodin jednoho počítače při běhu s různými operačními systémy, ovlivnění posunu hodin časovou synchronizací u systémů jiných než FreeBSD a neuvádění časových značek do hlaviček paketů TCP jakožto implicitní nastavení u systémů Windows. Pokud však počítač vykazuje konstantní posun, je možné ho identifikovat i po dlouhé uplynuté době (v řádu měsíců) a různých okolních podmínkách, jako jsou umístění, typ připojení či aktuální zatížení počítače.

Nástroj se podařilo nasadit na síťovou sondu studentských kolejí a měření zde budou pokračovat i v navazujícím výzkumu. Můžeme tak získat statistiky použitelnosti této techniky k identifikaci jednak koncových klientů, různých síťových zařízení a také internetových serverů poskytujících služby. S brzkým přechodem na protokol IPv6 může nástroj posloužit také k párování IPv4 a IPv6 adres patřících jednomu počítači.

V dalším vývoji programu by mohlo dojít k oddělení a vytvoření nástroje specializovaného na rozlišení počítačů za překladačem adres (NAT).

Výsledky této práce budou dále využity v rámci projektu *Moderní prostředky pro boj*

*s kybernetickou kriminalitou na Internetu nové generace.*

# Literatura


[1] Bayer, T.: Konvexní obálka množiny bodů. Algoritmy v digitální kartografii, Katedra aplikované geoinformatiky a kartografie, Přírodovědecká fakulta UK. Listopad 2010.

[2] Berners-Lee, T.; Fielding, R.; Frystyk, H.: Hypertext Transfer Protocol – HTTP/1.0. RFC 1945. Květen 1996.
URL http://tools.ietf.org/html/rfc1945

[3] Corporation, O.: Oracle VM VirtualBox: User Manual - Chapter 6. Virtual networking. 2004-2012.
URL http://www.virtualbox.org/manual/ch06.html

[4] Fielding, R.; Gettys, J.; Mogul, J.; aj.: Hypertext Transfer Protocol – HTTP/1.1. RFC 2616. Červen 1999.
URL http://tools.ietf.org/html/rfc2616

[5] Garcia, L. M.: Programming with Libpcap - Sniffing the network from our own application. *Hakin9*, ročník 3, č. 2, únor 2008: s. 38–46, ISSN 1733-7186.

[6] Graham, R. L.: An efficient algorithm for determining the convex hull of a finite planar set. *Information Processing Letters*, ročník 1, č. 4, leden 1972: s. 132–133, ISSN 0020-0190.

[7] Jacobson, V.; Braden, R.; Borman, D.: TCP Extensions for High Performance. RFC 1323. Květen 1992.
URL http://tools.ietf.org/html/rfc1323

[8] Kohno, T.; Broido, A.; Claffy, K.: Remote physical device fingerprinting. *Dependable and Secure Computing, IEEE Transactions on*, ročník 2, č. 2, květen 2005: s. 93–108, ISSN 1545-5971.

[9] Matoušek, J.; Gärtner, B.: *Understanding and using linear programming*. Universitext (1979), Springer, 2007, ISBN 978-3-540-30697-9, 222 s.

[10] McDanel, B.: TCP timestamping and remotely gathering uptime information. Březen 2001.
URL http://seclists.org/bugtraq/2001/Mar/182

[11] Mills, D.: Simple Network Time Protocol (SNTP) Version 4 for IPv4, IPv6 and OSI. RFC 4330. Leden 2006.
URL http://tools.ietf.org/html/rfc4330

[12] Mills, D.; Delavare, U.; Martin, J.; aj.: Network Time Protocol Version 4: Protocol and Algorithms Specification. RFC 5905. Červen 2010.
URL http://tools.ietf.org/html/rfc5905

[13] Mills, D. L.: Internet time synchronization: the network time protocol. *Communications, IEEE Transactions on*, ročník 39, č. 10, říjen 1991: s. 1482–1493, ISSN 0090-6778.

[14] Moon, S. B.; Skelly, P.; Towsley, D.: Estimation and removal of clock skew from network delay measurements. In *INFOCOM '99. Eighteenth Annual Joint Conference of the IEEE Computer and Communications Societies. Proceedings. IEEE*, ročník 1, březen 1999, s. 227–234.

[15] Murdoch, S. J.: Hot or not: Revealing hidden services by their clock skew. In *In 13th ACM Conference on Computer and Communications Security (CCS 2006)*, ACM Press, listopad 2006, s. 27–36.

[16] Murdoch, S. J.; Lewis, S.: Embedding covert channels into TCP/IP. *Information Hiding: 7th International Workshop*, ročník 3727, červenec 2005: s. 247–261.

[17] Postel, J.: Internet Control Message Protocol - DARPA Internet Program Protocol Specification. RFC 792. Září 1981.
URL http://tools.ietf.org/html/rfc792

[18] Postel, J.: Internet Protocol - DARPA Internet Program Protocol Specification. RFC 791. Září 1981.
URL http://tools.ietf.org/html/rfc791

[19] Postel, J.: Transmission Control Protocol - DARPA Internet Program Protocol Specification. RFC 793. Září 1981.
URL http://tools.ietf.org/html/rfc793

[20] Zalewski, M.: *Silence on the wire: field guide to passive reconnaissance and indirect attacks*. No Starch Press, Inc., první vydání, 2005, ISBN 15-932-7046-1, 281 s.

[21] Zander, S.; Murdoch, S. J.: An improved clock-skew measurement technique for revealing hidden services. In *Proceedings of the 17th conference on Security symposium*, USENIX Association, červenec 2008, s. 211–225.

# Příloha A

# Ukázka a popis konfiguračního souboru

```
############################
# Pcf konfigurační soubor #
############################


########################
# Nastavení filtrování #
########################

# Síťové rozhraní (eth0, wlan0, any, ...)
interface eth0

# Počet paketů (0 pro nekonečno)
num_packets 0

# Čas běhu (v sekundách, 0 pro nekonečno)
time 0

# Číslo portu (1-65535)
port 80

# Zdrojová adresa
src_address 192.168.1.2

# Cílová adresa
dst_address 192.168.1.217

# Pakety s nastaveným příznakem SYN (1 = ano, 0 = ne)
syn 1

# Pakety s nastaveným příznakem ACK (1 = ano, 0 = ne)
ack 0
```

```
# Další nastavení filtrů pcap
filter not host 192.168.1.1 and ether src 11:22:33:44:55:66


############
# Soubory #
############

# Soubor s aktivními počítači
active www/data/active.xml

# Soubor s databází uložených počítačů
database www/data/database.xml


########################
# Programové konstanty #
########################

# Práh pro rozlišení počítačů (1 ppm = 0.001)
THRESHOLD 0.001

# Počet paketů pro provedení blokových operací (výpočty, akt. souborů)
BLOCK 50

# Čas, po kterém bude neaktivní počítač odebrán (v sekundách)
TIME_LIMIT 3600
```

# Příloha B

# Schémata souborů XML sledovaných a uložených počítačů

## B.1 Schéma souboru XML se sledovanými počítači

Následující schéma specifikuje formát souboru právě sledovaných počítačů. Elementy „name“ a „diff“ jsou obsaženy v případě, že došlo k rozpoznání daného počítače. Vždy jsou uvedeny buď oba dva, nebo ani jeden, pokud počítač rozpoznán nebyl. Pokud došlo k identifikaci počítače s některým z počítačů uložených v databázi, obsahuje element „name“ jméno počítače získáné z databáze. Pokud došlo k rozpoznání dvou právě sledovaných zařízení, je jako název uložena síťová adresa rozpoznaného zařízení. Element „diff“ obsahuje hodnotu rozdílu posunů hodin identifikovaných počítačů.

```
<?xml version="1.0 encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">
 <xs:element name="computers">
  <xs:complexType>
   <xs:sequence>
    <xs:element name="computer" minOccurs="0" maxOccurs="unbounded">
     <xs:complexType>
      <xs:sequence>
       <xs:element name="name" type="xs:string" minOccurs="0" />
       <xs:elemetn name="diff" type="xs:double" minOccurs="0" />
       <xs:element name="address" type="xs:string" />
       <xs:element name="frequency" type="xs:integer" />
       <xs:element name="packets" type="xs:long" />
       <xs:element name="date" type="xs:string" />
      </xs:sequence>
      <xs:attribute name="skew" type="xs:double" use="required" />
     </xs:complexType>
    </xs:element>
   </xs:sequence>
  </xs:complexType>
 </xs:element>
</xs:schema>
```

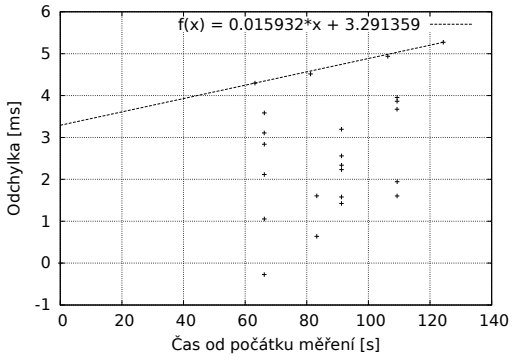
## B.2 Schéma souboru XML s uloženými počítači

Následující schéma specifikuje formát souboru uložených počítačů, jaký nástroj očekává.

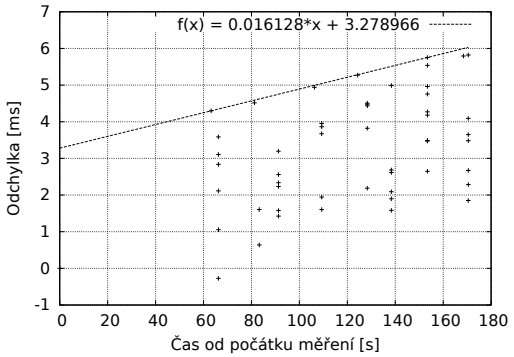
```
<?xml version="1.0 encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">
 <xs:element name="computers">
  <xs:complexType>
   <xs:sequence>
    <xs:element name="computer" minOccurs="0" maxOccurs="unbounded">
     <xs:complexType>
      <xs:sequence>
       <xs:element name="name" type="xs:string" minOccurs="0" />
       <xs:element name="address" type="xs:string" />
       <xs:element name="frequency" type="xs:integer" />
       <xs:element name="date" type="xs:string" />
      </xs:sequence>
      <xs:attribute name="skew" type="xs:double" use="required" />
     </xs:complexType>
    </xs:element>
   </xs:sequence>
  </xs:complexType>
 </xs:element>
</xs:schema>
```

# Příloha C

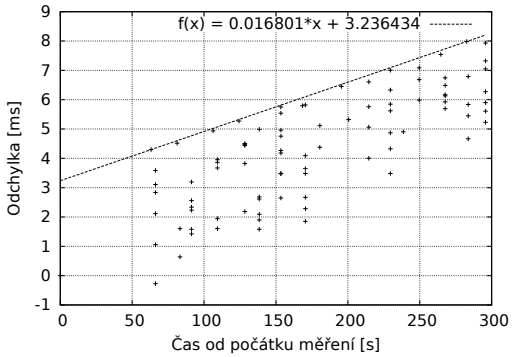
# Postup průběhu měření jednoho počítače



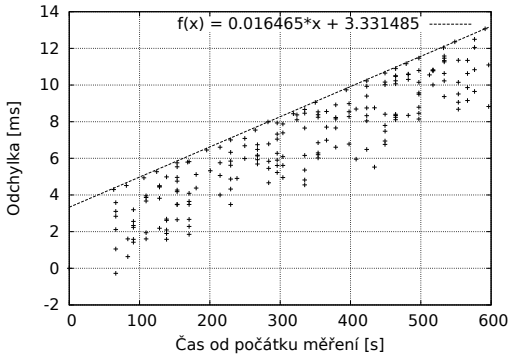

(a) Po 2 minutách




(b) Po 3 minutách

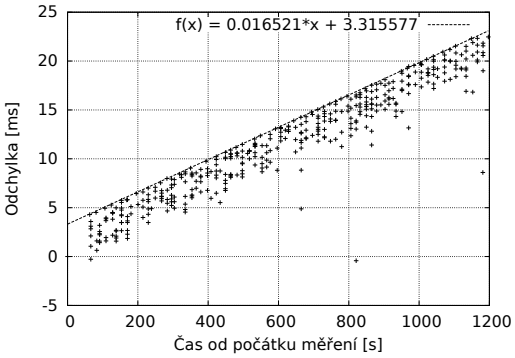



(c) Po 5 minutách

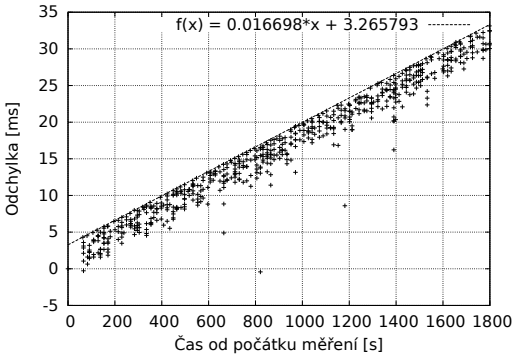



(d) Po 10 minutách

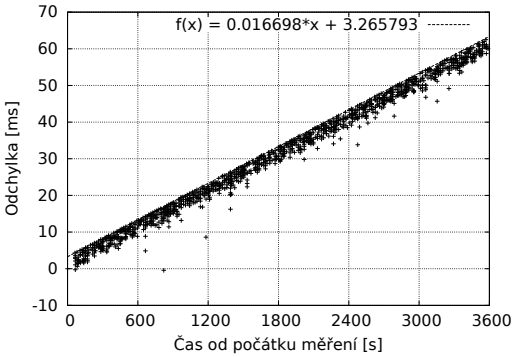
Obrázek C.1: Postup měření jednoho počítače — prvních 10 minut.



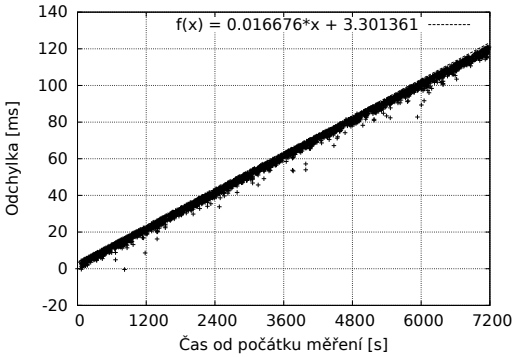

(a) Po 15 minutách




(b) Po 30 minutách




(c) Po 60 minutách




(d) Po 120 minutách

Obrázek C.2: Postup měření jednoho počítače — 15 až 120 minut.