VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV POČÍTAČOVÝCH SYSTÉMŮ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF COMPUTER SYSTEMS

URČOVÁNÍ IDENTITY POČÍTAČE POMOCÍ ODCHYLKY VNITŘNÍCH HODIN

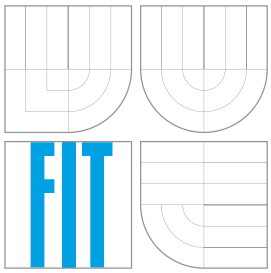
BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE
AUTHOR

BARBORA FRANKOVÁ

BRNO 2013

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV POČÍTAČOVÝCH SYSTÉMŮ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF COMPUTER SYSTEMS

URČOVÁNÍ IDENTITY POČÍTAČE POMOCÍ ODCHYLKY VNITŘNÍCH HODIN

COMPUTER IDENTITY BASED ON ITS INTERNAL CLOCK SKEW

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE — BARBORA FRANKOVÁ
AUTHOR

VEDOUCÍ PRÁCE — Ing. LIBOR POLČÁK
SUPERVISOR

BRNO 2013

Abstrakt

Tato práce se zabývá identifikací vzdálených počítačů na základě odchylky vnitřních hodin. Podává informaci o různých metodách sběru a vyhodnocení dat založených na časových razítkách získaných pomocí TCP, ICMP a Javascriptu. Součástí práce je balík software, který vybrané techniky uvádí do praxe. S jeho pomocí bylo provedeno velké množství experimentů, které se zabývaly působením různých faktorů na výsledky měření. Testován byl vliv software (operační systémy, prohlížeče a synchronizační nástroje) i hardware (teplota, typ napájení, vzdálenost a typ připojení k počítačové síti). Zajímavé výsledky přinesly zejména snahy o párování IPv4 a IPv6 adres a analýza vlivu NTP.

Abstract

Thesis deals with remote computer identification based on its internal clock skew. It describes various methods to collect and evaluate time-related data extracted from TCP, ICMP and Javascript. Software package that implements some of those methods is attached as well. During the work, many experiments were carried out to find out possible effects of certain factors. Tests were aimed on software (operation systems, web browsers and time synchronization) as well as hardware (temperature, power source, distance and network connection type) of the remote systems. Interesting results were discovered in areas of IPv4/IPv6 address pairing and NTP time synchronization.

Klíčová slova

identifikace počítače, vnitřní hodiny, posun hodin, odchylka hodin, časová razítka, TCP, ICMP, Javascript, synchronizace času, NTP, párování IPv4 a IPv6 adres

Keywords

computer identity, internal clock, clock skew, clock offset, timestamps, TCP, ICMP, Javascript, time synchronization, NTP, IPv4 and IPv6 address pairing

Citace

Barbora Franková: Určování identity počítače pomocí odchylky vnitřních hodin, bakalářská práce, Brno, FIT VUT v Brně, 2013

Určování identity počítače pomocí odchylky vnitřních hodin

Prohlášení

Prohlašuji, že jsem tuto bakalářskou práci vypracovala samostatně pod vedením pana Ing. Libora Polčáka.

.....
Barbora Franková
15. května 2012

Poděkování

Ráda bych poděkovala vedoucímu bakalářské práce, panu Ing. Liborovi Polčákovi, za odborné rady a připomínky.

Obsah

Seznam obrázků

Kapitola 1

Úvod

Jedním z rysů moderní společnosti je rychlý technologický pokrok v oblasti informačních technologií a výpočetní techniky. Tyto technologie se stávají standardem, který každodenně využíváme i k činnostem, které manipulují s více či méně chráněnými informacemi. Ruku v ruce s tímto pokrokem jdou také pokusy o nelegální získání soukromých informací. Počítačová kriminalita každým rokem roste a znalosti či schopnosti nutné ke kybernetickým útokům se stále snižují. Zatím co dříve musel být útočník víceméně profesionální programátor, v dnešní době nalezneme na Internetu programy a postupy, kterými je možné nelegálně získat velké množství informací.

Jednoduše zabránit počítačové kriminalitě není možné. Pachatelé za sebou nezanechávají fyzické stopy a ve velké míře zneužívají anonymitu, kterou Internet poskytuje. Mohou si připravit propracované a promyšlené útoky, kde omezením není ani geografická vzdálenost. Některé z útoků již byly zaneseny do zákona jako trestná činnost (např. neoprávněný přístup, odposlouchávání, narušení systému nebo počítačové padělání). Jednou z možností, jak rozšířit prostředky pro boj s počítačovou kriminalitou, je zlepšení identifikace pachatele.

Tato práce je zaměřena na identifikaci počítače v síti pomocí odchylky vnitřních hodin. Navazuje na diplomovou práci Jakuba Jiráska [4], v rámci které byl vytvořen program PC Fingerprinter. Cílem je provedení a zhodnocení sady experimentů, které zjistí ovlivnitelnost vypočítané odchylky vnitřním nastavením počítače i vnějšími podmínkami. Této problematice se jako první věnovali Kohno et al. [5], množství experimentů bylo provedeno i v dalších projektech [6, 7, 11] a v bakalářské práci Jana Novotného [9]. V programu PC Fingerprinter bylo provedeno několik úprav, z nichž nejdůležitější je rozšíření pro zpracování ICMP a Javscriptových časových známek.

Experimenty, které byly provedeny v rámci bakalářské práce, ověřily dopad velkého množství faktorů na výpočet odchylky hodin sledovaných zařízení. Výsledky testů zaměřených na sotfware prokazují velký vliv synchronizace hodin pomocí NTP. Výběr metody sběru dat ICMP, TCP nebo Javascript měl dopad na vypočítanou odchylku u většiny počítačů, především u zařízení s operačním systémem Mac OS X. Zajímavé výsledky přinesla analýza rozdílu posunu hodin mezi IPv4 a IPv6 adresami, kde se v laboratorních podmínkách podařilo spárovat většinu zařízení. Experimenty se zaměřením na hardware naopak nedokázaly potvrdit vliv teploty CPU, typu napájení a vzdálenosti sledovaných počítačů.

Princip identifikace zařízení pomocí odchylky vnitřních hodin je popsán v kapitole 2. Kapitola 3 se věnuje programu PC Fingerprinter a kapitola 4 rozšíření programu a vytvoření analytických nástrojů. V kapitole 5 jsou uvedeny experimenty, které byly v rámci bakalářské práce uskutečněny.

Kapitola 2

Identifikace počítače v síti

Veškerá zařízení v síti (koncová zařízení, směrovače, proxy servery apod.) obsahují vnitřní hodiny, které se skládají ze softwarových a hardwarových komponent [12]. Signál hodin je produkován krystalovým oscilátorem, který kmitá na určité frekvenci. Tato frekvence je závislá na parametrech výroby (např. na úhlu řezu) a na typu použitého krystalu. Mechanické nepřesnosti vznikající během výrobního procesu způsobují, že krystaly stejného typu, řady a výrobního data mohou mít trochu odlišné frekvence [6]. Frekvence vnitřních hodin může být ovlivněna i vnějšími faktory jako je okolní teplota nebo vlhkost [5].

Odchylka vnitřních hodin, kterou lze využít k identifikaci počítače, je dána hardwarovými a softwarovými vlastnostmi daného zařízení. Úmyslně modifikovat fyzické vlastnosti krystalu není v silách běžného uživatele a v tomto ohledu se proto jedná o relativně spolehlivou metodu pro identifikaci počítače. Na druhou stranu lze různě posunout a synchronizovat systémové hodiny, ze kterých se časové známky generují, což může identifikace zařízení naopak ztížit. Hlavním úkolem je ovšem získat tuto časovou informaci ze vzdáleného zařízení. K tomuto úkolu lze použít několik metod, které jsou popsány v sekcích 2.3 a 2.4.

2.1 Synchronizace času

Systémový čas nemusí být vždy úplně přesný. Za účelem synchronizace systémového času se spolehlivých zdrojem byly vytvořeny protokol NTP. Ten definuje přenos informací o aktuálním čase ke klientovi pomocí UDP paketů na transportní vrstvě a IP paketů na vrstvě síťové. Samotná synchronizace je zajištěna dočasným zrychlením nebo zpomalením systémových hodin. Odchylka systémového času od času na serveru se okamžitě po synchronizaci pohybuje v rozmezí 1-50 ms v závislosti na zdroji a kvalitě spojení [**?**].

Systémové hodiny mohou být synchronizovány jednorázově nebo pomoci démona. Pokud spustíme jednorázovou synchronizaci, nastaví se v daném okamžiku aktuální čas. Tento typ synchronizace je ve výchozím nastavení operačního systému Windows XP prováděn jedenkrát týdně. Démon (např. program ntpd) zajišťuje pravidelnou aktualizaci hodin. Vliv synchronizace na odchylku vnitřních hodin je u různých operačních systémů odlišný a je součástí experimentů popsaných v kapitole 5.

2.2 Aktivní a pasivní získávání časového razítka

K získání informace o čase ze vzdáleného počítače je nutné, aby sledující i sledované počítače byly součástí počítačové sítě. Pokud je tento předpoklad splněn, je možné využít více typů

metod. Jedná se o metody aktivní, pasivní a semi-pasivní. Hlavní rozdíl spočívá v roli sledovaného zařízení.

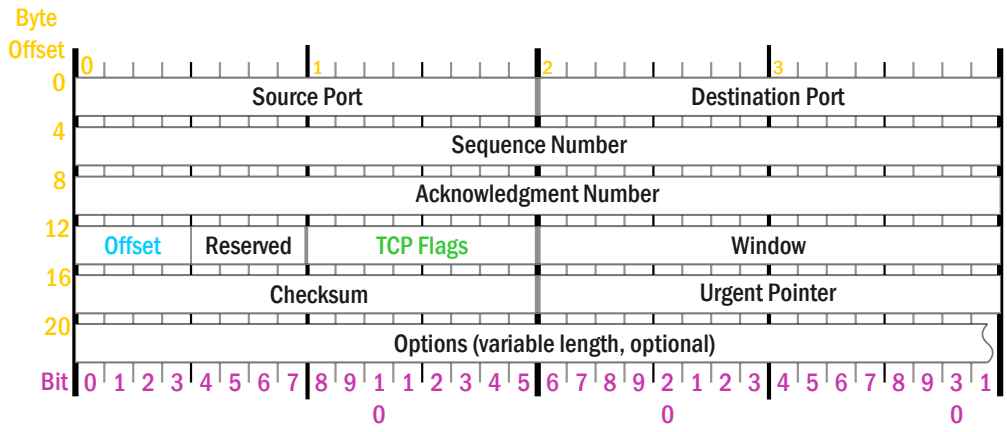
U aktivních technik je nutné vytvořit spojení mezi měřeným a měřícím zařízením, přičemž spojení inicializuje zařízení, které se snaží získat informace. Tyto techniky jsou viditelné ze strany sledovaného zařízení. Příkladem je zjišťování časového razítka ze zpráv ICMP [5], kdy sledovanému zařízení pošleme dotaz a následně čekáme na odpověď, nebo HTTP požadavky na webový server [12].

Pasivní metody také potřebují k zjišťování časové informace spolupráci vzdáleného zařízení, ale spojení je zahájeno sledovaným zařízením. Nespornou výhodou těchto technik je možnost jejich použití i v případě, že sledované zařízení se nachází za překladačem adres (NAT) nebo je chráněno firewallem [**?**]. Zároveň je odposlech dat neviditelný pro sledované zařízení. Příkladem může být zjišťování časové známky pomocí protokolu TCP.

Poslední možnou technikou je semi-pasivní získávání informací. Jedná se o kombinaci aktivního a pasivního přístupu. Tato metoda v sobě ukrývá výhodu pasivního přístupu, a sice že spojení je iniciováno sledovaným zařízením. Zařízení, které odposlouchává, může aktivně komunikovat se sledovaným zařízením přes vytvořené spojení a může si vyžádat informace, které potřebuje.

2.3 Získání časové informace pomocí TCP

Jednou z možností, jak získat časovou známku, je využití pole voleb (Options) v protokolu TCP. TCP je spojově orientovaný protokol, který pracuje nad transportní vrstvou a zajišťuje spolehlivý přenos dat. Obecná hlavička protokolu je uvedena na obrázku 2.1. RFC 1323 specifikuje možnosti použití TCP Timestamp Option (TSopt). Odesílané TCP segmenty obsahují časovou známku, pokud je TSopt implementován na obou stranách a zároveň je přítomen v iniciačním paketu s nastaveným příznakem SYN [5]. Hlavním úkolem TSopt je zlepšení přenosových vlastností protokolu[3].



Obrázek 2.1: Hlavička TCP segmentu.

Timestamp Option obsahuje dvě čtyřbytová čísla TSval a TSecr. TSval (Timestamp Value) je časová známka odeslání segmentu uvedená v milisekundách. Je generována pomocí

systémových hodin, což nám umožní určit odchylku a tedy i identifikovat zařízení. TSecr (Timestamp Echo Reply) je platná časová známka pouze v případě, že bit ACK v TCP hlavičce je nastaven na 1. Její hodnota odpovídá TSval potvrzovaného segmentu.

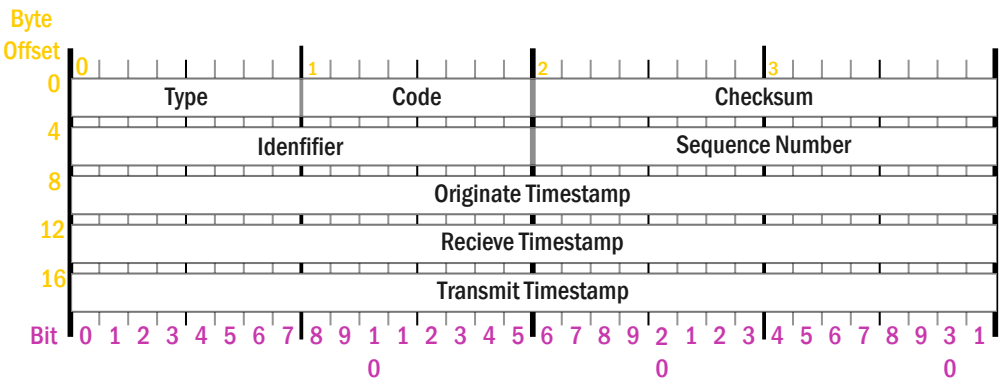
Nevýhodou toho přístupu je volitelnost implementace TSopt. Není pevně definována frekvence hodin, která závisí na operačním systému a hardwarové specifikaci počítače. Pohybuje se v rozmezí 2 Hz až 1 kHz [7]. K získání správného výsledku je nutné tuto hodnotu zpětně dopočítat.

2.4 Další možnosti získání časové informace

2.4.1 ICMP Timestamp

Internet Control Message Protocol (ICMP, RFC 792) byl vytvořen nad protokolem IP za účelem zefektivnění komunikace na počítačové síti. Obecný formát hlavičky ICMP pro IPv4 obsahuje typ zprávy, její kód a kontrolní součet. Časová značka je součástí zprávy typu 13 s kódem 0 (Timestamp). Tato zpráva je odeslána sledovanému zařízení a poté se čeká na odpověď. Odpovědí by měla být ICMP zpráva typu 14 s kódem 0 (Timestamp Reply), která je znázorněna na obrázku 2.2. ICMP odpověď obsahuje tři 32bitové hodnoty:

- *Originate Timestamp* – časová známka původní ICMP zprávy
- *Recieve Timestamp* – čas, kdy byla odpovídající ICMP zpráva přijata
- *Transmit Timestamp* – čas, kdy byla odeslána odpověď



Obrázek 2.2: ICMP zpráva typu 14 s kódem 0.

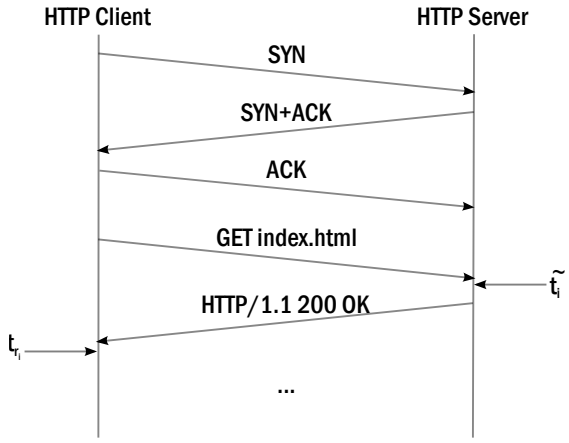
Poslední hodnota je pro nás relevantní. Obsahuje čas v milisekundách, který uběhl od půlnoci v odpovídajícím časovém pásmu do chvíle, kdy byl paket odeslán. Hodnota je získána ze systémových hodin sledovaného zařízení, což nám umožní vypočítat odchylku vnitřních hodin.

Výhodou tohoto přístupu je pevná frekvence 1 kHz. Hlavní nevýhodou je nutnost aktivní spolupráce sledovaného zařízení, které nesmí být umístěno za překladačem adres (NAT) nebo za firewallem filtrujícím ICMP zprávy. ICMP zprávy přitom nemusí být filtrovány

pouze cílovým zařízením, ale jakýmkoliv směrovačem po cestě. Další nevýhodou je nemožnost použití ICMPv6 zpráv, které nemají definovaný ekvivalent k ICMP Timestamp a ICMP Timestamp Reply.

2.4.2 HTTP

Další možností, jak získat časovou známku, je využití protokolu HTTP. Této variantě se věnovali Zander et al. [12]. Sledující počítač se chová jako klient a po navázání spojení odesílá HTTP požadavky (např. GET) na sledovaný webový server. Tato komunikace je znázorněna na obrázku 2.3. Standardní servery generují časové známky na frekvenci 1 Hz a uvádí tuto hodnotu v odpovědi v poli Datum (Date). Pro HTTP 1.0 je použití těchto známek doporučeno, od HTTP 1.1 je jejich uvedení povinné s výjimkou některých 5xx chybových zpráv a 1xx odpovědí.



Obrázek 2.3: HTTP zprávy zahrnující požadavek a odpověď, které potřebujeme k zjištění časové známky.

Tento přístup je zaměřen pouze na servery a vyžaduje aktivní spolupráci sledovaného zařízení. Zároveň je problémem nízká frekvence hodin, neboť může dojít k tzv. chybě kvantování, která může dosáhnout až jedné sekundy [12]. Jakékoliv přesnější měření je potom znemožněno.

2.4.3 Sekvenční čísla TCP

Možnosti získání časové známky pomocí TCP Timestamp jsou popsány v sekci 2.3. Časovou známku z TCP segmentu lze u některých operačních systémů zjistit i pomocí sekvenčních čísel. Sekvenční čísla jsou 32bitové hodnoty, které se primárně využívají pro zajištění spolehlivého přenosu. Uvedení tohoto čísla v poli Acknowledgement Number znamená úspěšné doručení všech segmentů, které měly menší sekvenční číslo.

Vytvoření ISN (Initial Sequence Number) je záležitostí operačního systému. Pro operační systém Linux je způsob generování ISN popsán v [8]. K výpočtu ISN se využívá kryptografický algoritmus (SHA-1, MD4 nebo MD5) a aktuální systémový čas s frekvencí 1 MHz. Výsledky měření odchylky hodin jsou poměrně přesné díky vysoké frekvenci hodin,

jenže kryptografický klíč se mění každých 5 minut. Výpočet odchylky hodin během delší doby je tím pádem poměrně složitý.

2.4.4 Javascript

Další možností jak získat časovou známku sledovaného počítače s využitím protokolu HTTP je Javascript. Ten by měl být podporován ve většině používaných prohlížečů. Vhodně sestavený skript může odesílat v pravidelných intervalech časovou známku zjištěnou funkcí getTime pomocí HTTP metody GET nebo POST. Frekvence časových známek je 1 kHz, neboť Javascript je schopný vygenerovat čas na úrovni milisekund [2].

Výhodou tohoto přístupu je velká pravděpodobnost, že sledovaný počítač bude mít Javascript povolený i ve výchozím nastavení. Narozdíl od TCP, kde je implementace TSopt volitelná, je možné pomocí Javascriptu získat časovou známku nezávisle na operačním systém, prohlížeči i síťové topologii. Frekvence je stejně jako u ICMP pevně daná a nezávislá na operačním systému. Rozdílem oproti ostatním metodám je formát časové známky. TCP i ICMP odesílají 32bitovou hodnotu, která odpovídá počtu hodin, minut, sekund a milisekund (v závislosti na operačním systému) od půlnoci daného dne. Javascript vrací UNIXovou časovou značku, jejíž hodnota značí počet milisekund od 1. 1. 1970.

2.5 Princip identifikace pomocí odchylky hodin

V této i následujících kapitolách budeme používat terminologii zavedenou v [4]. Definujme hodiny C, které počítají čas uběhlý od určitého počátečního času. Počáteční čas závisí na typu zařízení (např. operační systémy Unix používají 1. 1. 1970 [6]). Hodnota $r[C]$ reprezentuje nejmenší možný časový úsek, o který lze časovou informaci navýšit. Frekvence hodin C je obrácená hodnota $r[C]$. Běžně se frekvence hodin pohybuje v rozmezí 2 až 100 Hz.

Hodnota $off[C]$ je odchylka hodin v milisekundách, která odpovídá rozdílu mezi reálným a zjištěným systémovým časem. Je to funkce, která s časem roste nebo klesá. Pokud budeme předpokládat, že vypočítaná odchylka hodin je spojitá funkce s nezávislou proměnnou t, můžeme definovat posun vnitřních hodin zařízení $s[C]$ jako první derivaci odchylky podle času. Posun je v čase konstantní a jednotkou jsou milisekundy za sekundu (ms/s) nebo ekvivalentní počet částí v milionu (ppm), kde 0,001 ms/s odpovídá 1 ppm [4].

2.6 Výpočet posunu a odchylky hodin

Výpočet posunu vnitřních hodin je založen na principu, který je podrobně popsán v [5, 4]. Předpokladem je přijetí dostatečného počtu časových známek ze vzdáleného počítače se správně nastavenou časovou známkou. Nejdříve vypočítáme relativní čas uběhlý od počátku měření po doručení paketu a rozdíl hodnoty časového razítka daného paketu a časového razítka prvního paketu. Dále spočítáme frekvenci hodin a s jejím využitím upravíme hodnotu rozdílů časových razítek. Odchylka paketu je dána jako rozdíl mezi časem udaným časovým razítkem a časem reálným.

Dopočítání posunu vnitřních hodin lze provést dvěma způsoby. První z nich je metoda založená na lineárním programování [1]. Potřebujeme-li zjistit rychlost změny odchylek, proložíme body lineární funkcí, která shora ohraničuje množinu všech odchylek, a změříme úhel, který tato přímka svírá s osou x [4]. Druhou možností je použití metody nejmenších

čtverců. Principem je proložení bodů přímkou, pro kterou platí, že součet druhých mocnin vertikálních vzdáleností všech bodů je nejmenší možný.

Kapitola 3

Program PC Fingerprinter

Nástroj PC Fingerprinter (PCF), který je schopný na základě síťové komunikace nad protokolem TCP identifikovat počítač, byl vytvořen v rámci [4]. Tento nástroj ve formě démona naslouchá na síťovém rozhraní a podle nastavení v konfiguračním souboru analyzuje veškerou komunikaci. Zúčastněné počítače a naměřené hodnoty jsou ukládány do výstupního XML souboru. Ke každému ze sledovaných počítačů je průběžně generován graf odchylky jeho vnitřních hodin ve formátu SVG. Rozšíření programu a vytvoření komplexnějších analytických nástrojů v rámci bakalářské práce je popsáno v kapitole 4.

Nástroj je implementován v jazyce C++ a určen pro operační systém Linux. Instalaci programu lze provést pomocí příkazů `make` a `make install`. K překladu programu jsou nutné knihovny *libpcap* a *libxml2*. Pro zobrazování grafů vyžaduje PCF program *Gnuplot*.

3.1 Běh programu

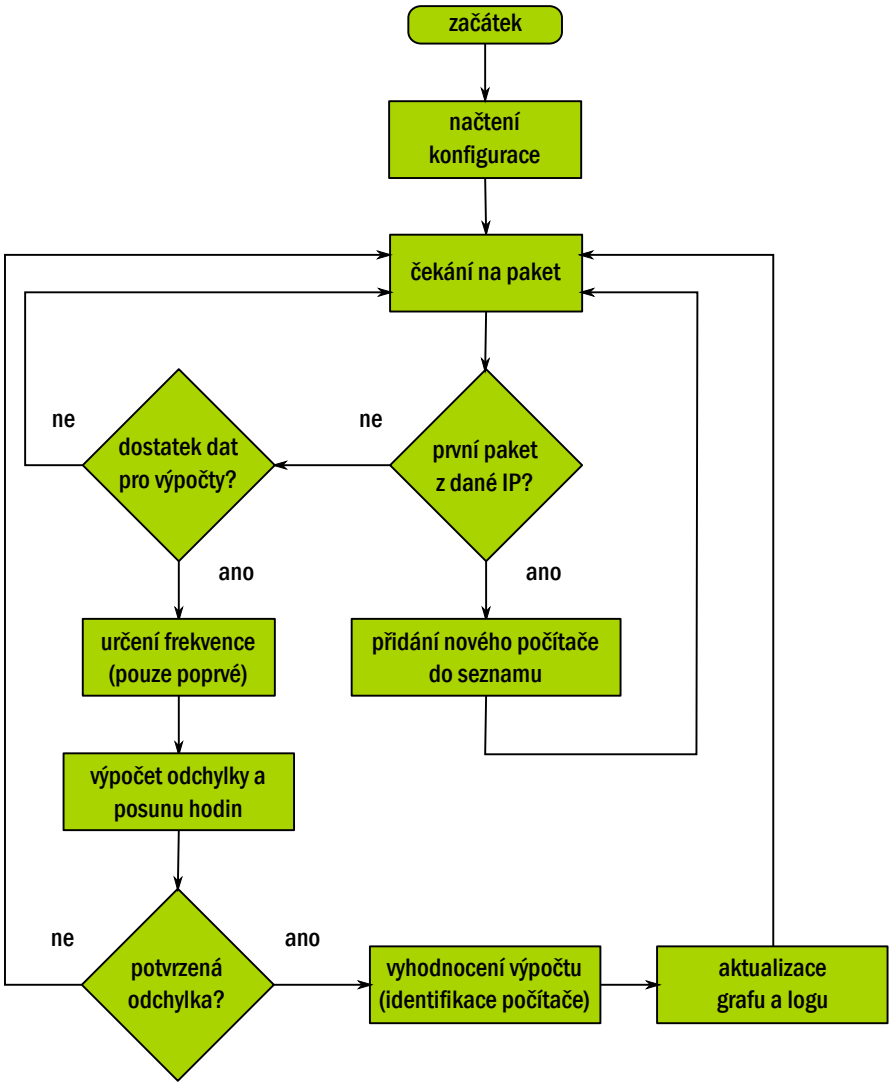
Vývojový diagram, který popisuje běh programu, je uveden na obrázku 3.1. Při spuštění nástroj PCF nejdříve zpracuje konfigurační soubor a následně vstoupí do smyčky, ve které čeká na pakety. Nově přijaté pakety jsou na základě zdrojové IP adresy roztříděny a uloženy spolu s informací o čase přijetí a časové známce, kterou program z paketu získá. Pokud došlo k přijetí dostatečného počtu paketů ze sledované adresy, je spuštěno blokové zpracování paketů vedoucí na výpočet frekvence (pouze jedenkrát pro daný počítač) a odchylek vnitřních hodin. Na základě těchto informací se program pokusí daný počítač identifikovat. Zároveň je vygenerován nebo aktualizován graf a změny jsou uloženy do odpovídajících souborů. Poté se běh programu vrátí zpět do smyčky, ve které čeká na další pakety.

3.2 Implementace

Program PCF vyžaduje ke svému chodu knihovnu *libpcap*. Ta je primárně určena pro UNIXové operační systémy a nabízí přímý přístup k paketům na jakémkoliv rozhraní počítače. Umožňuje aplikaci komplexních filtrovacích pravidel. Pomocí knihovny *libpcap* získáme čas přijetí paketu od počátku měření. Samotné časové razítko, které PCF potřebuje k výpočtu odchylky a posunu vnitřních hodin pak získáme z hlavičky paketu.

Veškeré hodnoty potřebné k výpočtu odchylky a identifikaci počítače uchovává třída `ComputerInfo`. Každá instance této třídy odpovídá jedné IP adrese a jsou zde uvedeny následující údaje:

- síťová adresa



Obrázek 3.1: Vývojový diagram popisující průběh programu PCF.

- seznam přijatých paketů
- vypočítaná frekvence
- potvrzená odchylka vnitřních hodin

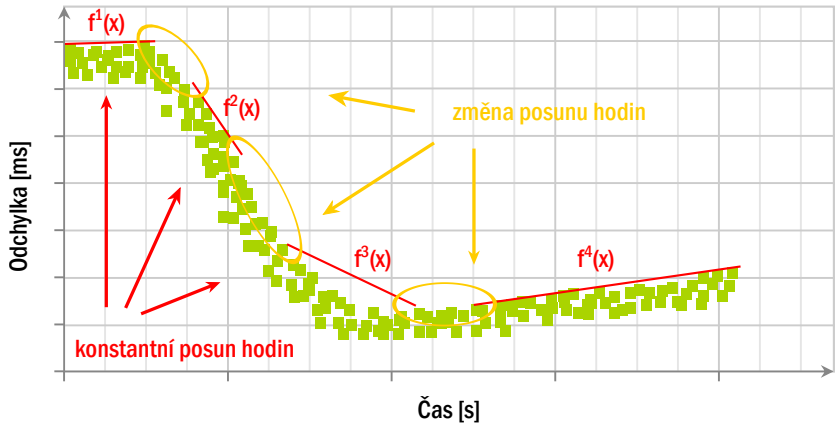
Instance třídy `ComputerInfo` jsou uloženy v seznamu ve třídě `ComputerInfoList`. Počítače s podobnou odchylkou se hledají jak v tomto seznamu, tak v XML databázi. Předávání dat mezi částmi programu, které zajišťují výpočet, porovnávání odchylek a aktualizaci grafů, probíhá pomocí třídy `AnalysisInfo`. Ta obsahuje následující údaje:

- IP adresu daného počítače
- seznam IP adres počítačů s podobnou odchylkou

- seznam úseček vypočítaných odchylek k vykreslení grafu

Pro určení posunu hodin sledovaného počítače je třeba získat rovnici přímky, která co nejtěsněji ohraničuje bodu reprezentující odchylky přijatých paketů. Odchylka každého paketu a výpočet frekvence je proveden podle postupu uvedeného v kapitole 2. Následně je využit algoritmus Graham Scan způsobem uvedeným v [5, 7] pro nalezení horní konvexní obálky. Ta v našem případě obsahuje jedinou úsečku, která shora ohraničuje přijaté pakety. Sklon úsečky odpovídá hledanému posunu.

Program PCF [1] v případě potřeby vypočítá úseček několik. Příklad takové situace je uveden na obrázku 3.2. Pakety jsou zpracovávány po blocích, jejichž velikost je daná konfiguračním souborem, nebo po uplynutí doby 5 minut. Pokud se nově vypočítaná hodnota posunu hodin liší od předchozího výsledku o více než stanovenou konstantu (ve výchozím nastavení tato hodnota odpovídá 10 ppm), předpokládáme, že došlo k určité změně v průběhu časových známek (např. synchronizace času pomocí nástroje *ntpdate*). Pětiminutový úsek, ve kterém byla detekována změna, se při výpočtu ignoruje. Do grafu je v takovém případě vložena pětiminutová mezera a nová rovnice přímky, která se bude brát jako potvrzená odchylka, se vypočítá po uplynutí dalších pěti minut. Tyto mezery jsou také vidět na obrázku 3.2.



Obrázek 3.2: Vypočtené funkce ohraničující měnící se odchylku hodin.

Součástí balíčku je také nástroj `log_reader`, který byl vytvořen pro zpětné nebo opětovné zpracování souborů z logu. Na rozdíl od programu PCF nevypočítá odchylku z časových údajů získaných z paketů v reálném čase, ale zpracuje soubor, ve kterém jsou uloženy časové údaje dříve přijatých paketů. Výsledkem je stejně jako u PCF graf s vypočtenými odchylkami hodin.

3.3 Uživatelské rozhraní a výstupy programu

Všechny aktivní počítače jsou uloženy v XML souboru. Obsah tohoto souboru se dynamicky mění podle aktuálního stavu komunikace. Uživatel má možnost uložit jakýkoliv počítač do

[1] `https://github.com/polcak/pcf`

databáze, která zůstává perzistentní i po ukončení programu. Identifikace počítače potom probíhá mezi aktivními počítači i počítači v databázi. Pokud při porovnání dojde ke shodě mezi více počítači, je vybrán ten s nejbližší hodnotou posunu vnitřních hodin.

Průběh výpočtu může uživatel ovlivnit pomocí konfiguračního souboru. V něm může nastavit například proměnnou, která určuje počet paketů nutných ke spuštění blokového výpočtu a aktualizace grafů. Může také nastavit rozhraní a port, na kterém bude program naslouchat, cílový počet zachycených paketů nebo délku měření, zdrojovou a cílovou IP adresu.

K ulehčení práce s programem bylo vytvořeno grafické uživatelské rozhraní s využitím HTML, PHP, Javascriptu a CSS. To umožňuje přehledně zobrazit zjištěné informace o sledovaných zařízeních, jako jsou například počet zpracovaných paketů, časy přijetí prvního a posledního paketu, naměřený posun hodin nebo graf ve formátu SVG.

Kapitola 4

Rozšíření analytických nástrojů

Součástí experimentů navržených v kapitole 5 je porovnání odchylek vypočítaných z TCP, ICMP a Javascriptu. Z tohoto důvodu byl program PCF rozšířen tak, aby bylo možné zpracovávat časová razítka z různých zdrojů. Implementace tohoto rozšíření je popsána v sekci 4.1. K rozsáhlejší analýze výsledků bylo vytvořeno několik podpůrných skriptů, které jsou uvedeny v sekci 4.2.

4.1 Rozšíření programu PCF

V rámci bakalářské práce byla navržena úprava zdrojových kódů umožňující zpracování časových razítek z více údajů. Ta byla následně implementována společně s novými metodami získávání časových známek pomocí ICMP a Javascriptu[1]. Na straně serveru, ke kterému sledované počítače přistupují, byl nově doplněn skript, který v pravidelných intervalech odesílá časové známky získané pomocí metody `getTime`. Tyto známky jsou pak odesílány HTTP metodou GET na IP adresu serveru.

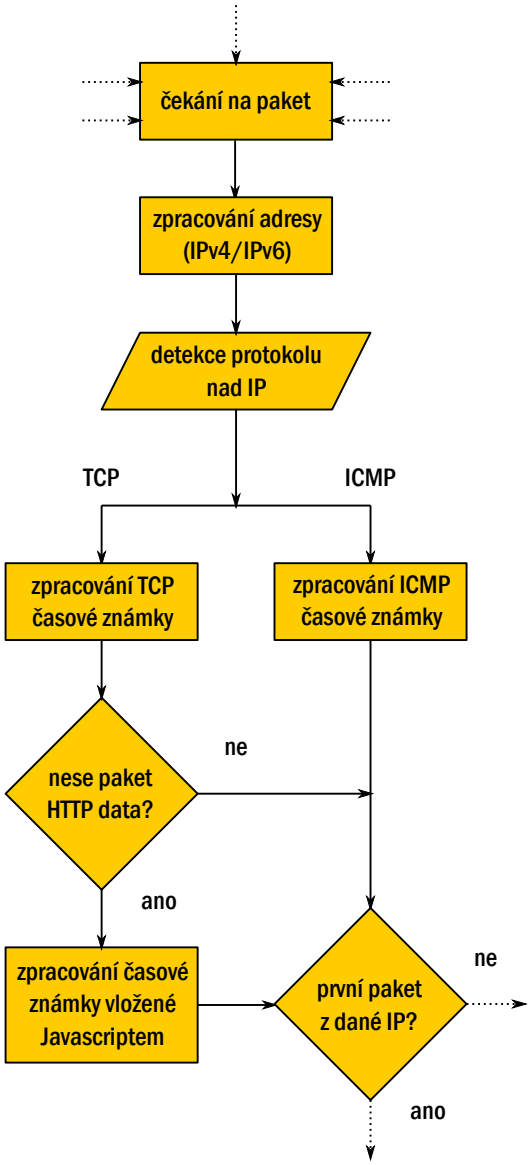
Základní běh programu zůstal v původní podobě, nicméně byly upraveny některé třídy, aby bylo možné odlišit časové známky počítačů získané z různých typů paketů. Na začátku běhu programu jsou vytvořeny tři instance třídy `ComputerInfoList` pro jednotlivé typy paketů (TCP, ICMP, Javascript). Ty pak obsahují seznam instancí třídy `ComputerInfo`, přičemž každá z nich uchovává informace o jednom sledovaném počítači. Seznamy TCP a Javascript mají podobný charakter, protože hodnotu časové známky získávají ze stejného paketu. Seznam počítačů v ICMP je odlišný a je proto tvořen třídou `ComputerInfoIcmp`, která je dědí ze třídy `ComputerInfo` a obsahuje navíc ukazatel na vlákno, které se využívá k odesílání ICMP zpráv.

Při přijetí paketu se nejdříve zjistí, zda jde o TCP nebo ICMP. Pokud se jedná o TCP paket odeslaný z IP adresy, která ještě není v seznamu, vloží se do seznamu nový počítač včetně informace o prvním paketu. Zároveň se zavolá funkce, která ověří, zda se na danou IP adresu mohou začít odesílat ICMP zprávy, a vytvoří se nové vlákno, které v pravidelných intervalech začne tyto zprávy odesílat nezávisle na běhu programu. V případě, že počítač je nalezen v seznamu zařízení sledovaných metodou TCP, program nově zjišťuje, zda se na aplikační vrstvě vyskytuje HTTP požadavek obsahující časovou známku vygenerovanou Javascriptem. Celý mechanismus je znázorněn na obrázku 4.1. Hodnota časové známky Javascriptu je vyjádřena v jiných jednotkách než TCP a ICMP, takže před samotným

[1] `https://github.com/bfrankova/pcf`

přidáním paketu a případným výpočtem nového posunu je převedena z UNIXové známky určující počet milisekund od 1. 1. 1970 na počet milisekund od půlnoci.

V rámci úprav kódu byla přidána také globálně dostupná třída `Configurator`, která obsahuje konstanty z konfiguračního souboru a některé parametry příkazové řádky. Tato třída byla vytvořena pomocí návrhového vzoru *Jedináček (Singleton)*.



Obrázek 4.1: Detail zpracování nově přijatého paketu.

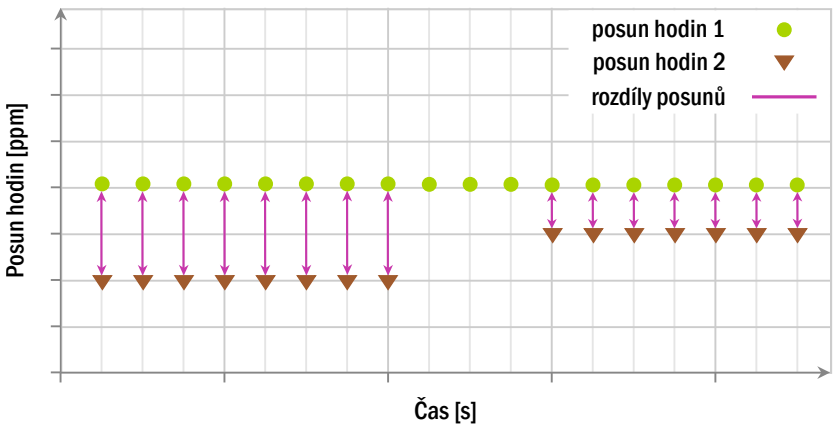
4.2 Vyhodnocovací skripty

Nově vytvořené skripty pro vyhodnocení výsledků měření jsou tři: *bulk_log_reader.sh*, *coupler.py* a *match_finder.sh*. Umožňují pokročilejší analýzu dat, než jaká je k dispozici při běhu programu PCF. Z důvodu vysoké výpočetní náročnosti byly skripty vytvořeny jako nezávislé na programu PCF tak, aby nezatěžovaly procesor během sběru dat. Při měření velkého množství počítačů byly pozorovány problémy i při porovnávání standardní metodou a při vykreslování grafů, takže komplexní hledání zařízení s podobnou odchylkou je vhodné provést zpětně s využitím logů programu PCF. Zároveň je možné provést analýzu dat z více běhů programu, což se hodí zejména pro opakované experimenty nebo pro experimenty měřené s velkým časovým odstupem.

Vhodným využitím pro celou sadu programů by mohla být například implementace PCF do síťové sondy, která by pouze sbírala data. Ve stanovených intervalech by pak všechny logy odesílala analyzátoru, který by provedl komplexní porovnání, vykreslení grafů a zpracování všech výsledků. V rámci bakalářské práce byl tento proces realizován ručně.

Skript *bulk_log_reader.sh* byl vytvořen za účelem automatizace zpracování logů a pouze hromadně zpracuje všechny soubory s uloženými časovými údaji o přijatých paketech, vytvoří skripty pro *gnuplot* a odpovídající grafy. Skript je určen pro UNIXový shell *bash*.

Skript *coupler.py*, určený pro Python 3, zpracovává skripty pro *gnuplot*, v nichž jsou ke každému počítači uloženy funkce reprezentující vypočítané odchylky hodin ve formátu $f_n(x) = a \cdot x + b$. Ke každé funkci je zároveň uveden interval, pro který je daná funkce platná. Skript *coupler.py* zpracuje blok funkcí pro daný počítač a posuny hodin a převede na diskrétní veličinu. Pro každý okamžik měření tak získáme hodnotu aktuálního posunu hodin, který je převeden z ms/s na ppm. Pokud pro určitý bod na ose x neexistuje funkce, hodnota posunu je nastavena na *nan*. Skript *coupler.py* zpracovává vždy dvojici souborů, z nichž vypočítá rozdíl posunů hodin ke každému bodu na ose x (znázorněno na obrázku 4.2). Na základě délek překrývajících se segmentů vypočítáme vážený průměr ze všech hodnot rozdílů posunů a tuto hodnotu použijeme jako výsledek. Čím je hodnota menší, tím považujeme porovnávané počítače za podobnější.



Obrázek 4.2: Znázornění diskretizace posunů hodin a výpočtu rozdílu mezi dvěma vstupními soubory.

Skript *match_finder.sh* určený pro UNIXový shell *bash* pak umožňuje aplikaci skriptu *coupler.py* na všechny dvojice počítačů. Výstupem tohoto skriptu je přehledná tabulka v textové podobě.

Kapitola 5

Experimenty

Návrh experimentů, jejich realizace a vyhodnocení je hlavní součástí této práce. Experimenty byly navrženy tak, aby v co největší míře otestovaly výpočet odchylky vnitřních hodin v závislosti na nastavení počítače i okolních podmínkách. Testování bylo provedeno s odpovídajícím množstvím sledovaných počítačů tak, aby bylo možné prokázat cíl měření. Experimenty jsou rozděleny do dvou kategorií, v sekci 5.2 jsou testy zaměřené na software a v sekci 5.3 na hardware.

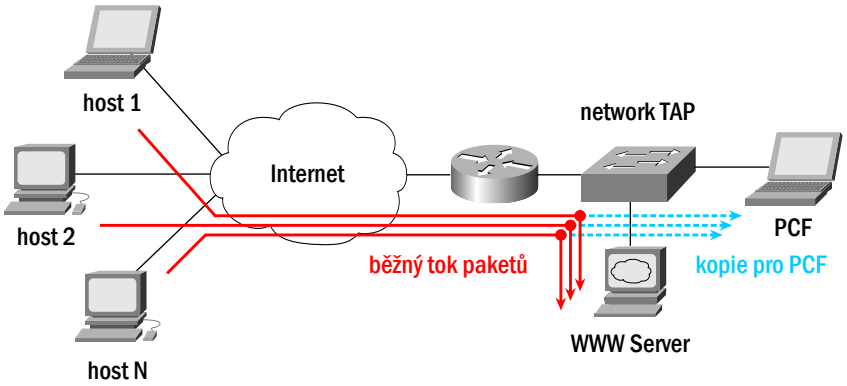
5.1 Příprava

Pro jednotný výpočet odchylky nezávisle na typu připojení nebo umístění měřeného a měřícího počítače byl uveden do provozu server s veřejnou IPv4 a IPv6 adresou. Počítače přistupovaly na stránku, která v pravidelných intervalech odesílala na server pakety s časovou známkou v TCP hlavičce a Javascriptovou známkou v HTTP požadavku. Veškerá komunikace serveru byla přeposílána na měřící počítač pomocí hardwarového přepínače nebo programu *iptables*. Ani jeden z těchto přístupů nezasahoval do struktury paketu, takže hodnoty časových známek byly doručeny programu PCF v nezměněné podobě. Jako měřící počítač byl zvolen počítač s operačním systémem Ubuntu 12.04, který zůstal v rámci všech testů stejný. Všechny naměřené výsledky jsou relativní, protože měřící počítač má vlastní odchylku vnitřních hodin. Měřením se stejným počítačem ale snížíme vliv použitého přístroje na interpretaci výsledků na minimum. Použitá topologie je znázorněna na obrázku 5.1.

5.1.1 Zjištěné problémy

Během přípravy na měření byly zjištěny určité komplikace. První z nich (uvedena v [4]) je nutnost úpravy registrů u počítačů s operačním systémem Windows tak, aby odesílaly časovou známku v TCP hlavičce. Ačkoliv je v RFC 1323 doporučována implementace časové známky, Windows ve výchozím nastavení známku neodesílají a činí tak až po přidání řetězce *Tcp1323Opts* s hodnotou REG_DWORD nastavenou na „2“ nebo „3“ v klíči registru

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



Obrázek 5.1: Topologie sítě využitá při měření odchylky hodin počítačů.

Druhou možností, jak povolit odesílání časové známky v TCP hlavičce, je zadání příkazu

```
netsh int tcp set global timestamps=enabled
```

do příkazové řádky spuštěné s administrátorskými právy (nutné pro časové známky v IPv6). Fakt, že Windows neodesílají TCP známky ve výchozím nastavení, zkomplikoval veřejné testy na porovnání TCP a Javascriptu, které jsou uvedeny v sekci 5.2.1. U všech testovaných verzí operačního systému Windows byl také zjištěn problém se získáváním dat pomocí ICMP, protože časová známka byla doručena s obráceným pořadím bytů. Za běhu programu nedokážeme identifikovat opračí systém sledovaného zařízení, takže podpora získávání ICMP dat z Windows není možná.

Dalším problémem je nemožnost získání časových známek ICMP z operačního systému Mac OS X. ICMP požadavek zůstává bez odpovědi. Podle neoficiálních serverů lze usoudit, že tento operační systém již nemá naimplementovanou podporu pro tento typ ICMP zpráv. Operační systémy spolu s metodami, které lze použít k získání časových známek, jsou uvedeny v tabulce 5.1.

5.1.2 Minimální délka měření

Podle výsledků měření v [5] bylo zjištěno, že k výpočtu odchylky vnitřních hodin počítače je dostatečná doba měření kolem jedné hodiny. Po uplynutí této doby se výsledky lišily jen minimálně. Cílem prvního experimentu, který je spíše přípravou pro následující experimenty, je ověřit toto tvrzení a pokusit se zjistit, jaký je minimální možný počet paketů a minimální doba měření na určení odchylky.

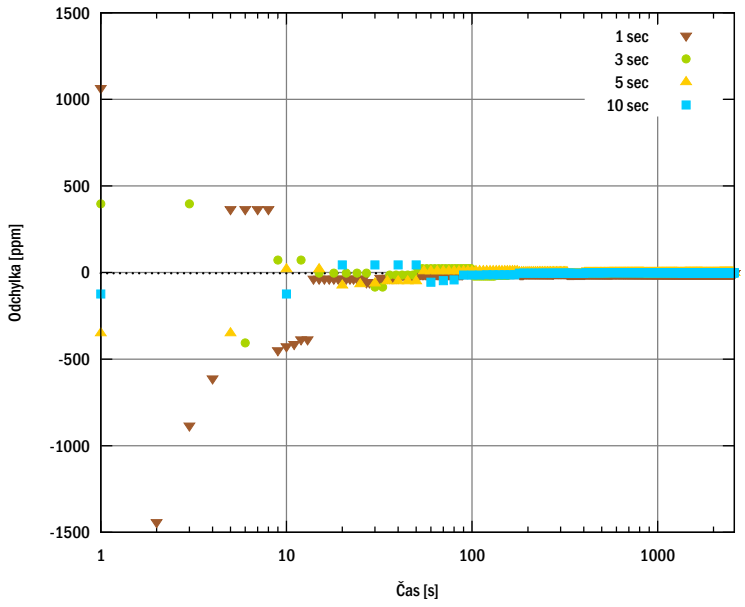
Experiment proběhl s počítačem, který měl nainstalován operační systému Ubuntu 12.04. Tento operační systém byl vybrán z důvodu stability odchylky i během dlouhodobých měření. Program PCF byl pro tento test lehce upraven a vypisoval vypočtený posun hodin

Operační systém	TCP	ICMP	Javascript
Windows	ano (nutná úprava registru)	**ne** (nestandardní)	ano
Ubuntu	ano	ano	ano
FreeBSD	ano	ano	ano
Mac OS X	ano	**ne**	ano
Android	ano	ano	ano

Tabulka 5.1: Přehled možností získání časových známek pro jednotlivé operační systémy.

po každém přijatém paketu a nepočítal frekvenci, která byla nastavena na počátku programu na odpovídající hodnotu 250 Hz. Na straně serveru byly mezi jednotlivými měřeními provedeny úpravy skriptu tak, aby se měnil interval mezi jednotlivými pakety odesílanými sledovaným zařízením na 1, 3, 5 a 10 sekund.

V grafu 5.2 jsou uvedeny výsledky prvního měření, kde se posun hodin přepočítával po každém přijatém paketu. Při velmi malém rozestupu mezi pakety jsou vypočítané hodnoty z prvních paketů poměrně nepřesné. Z grafu lze usoudit, že posun hodin se výrazně nemění po uplynutí zhruba 200 sekund od počátku měření. Naměřené hodnoty se pohybovaly v rozmezí od -5 do 1 ppm, tak velký rozptyl ale mohl být způsoben příliš častým přepočítáváním posunu.



Obrázek 5.2: Závislost posunu hodin na délce měření. Výpočet nové hodnoty probíhal po každém přijatém paketu).

Přesné hodnoty posunů hodin naměřené na stejném zařízení jsou uvedeny v tabulce 5.2. Jedná se opět o závislost posunu hodin na počtu přijatých paketů a délce měření. V tomto

případě probíhal výpočet standardní cestou vždy po přijetí bloku deseti paketů s časovou známkou. Přibližný výsledek byl spočítán po 8 minutách.

Interval mezi pakety	10 s	100 s	500 s	1000 s	3600 s
1 s	-220,947	-349,854	-353,157	-353,574	-353,708
3 s	-543,378	-356,631	-353,072	-353,737	-353,702
5 s	-351,639	-354,774	-352,983	-353,487	-353,682
10 s	-347,904	-351,846	-353,004	-353,808	-353,695

Tabulka 5.2: Závislost posunu hodin na počtu přijatých paketů a délce měření (v ppm).

Výsledky tohoto experimentu se přibližně shodovaly se závěry, ke kterým došli v [11]. Při práci stanovili minimální možný počet paketů pro výpočet odchylky na 70, což odpovídá necelým šesti minutám měření s pětisekundovým rozestupem mezi pakety. Naopak v práci J. Novotného [9] se k určení odchylky používalo v rámci jednoho měření více než 1 milion paketů. Intervaly mezi pakety musely být velmi malé, protože celková délka měření se pohybovala od 5 do 10 minut. Je tedy možné, že program byl nepřiměřeně paměťově i výpočetně náročný.

V případě, že ale budeme chtít získat odchylku na úrovni mikrosekund z časových známek s frekvencí 1 kHz, potřebujeme délku měření alespoň 1000 sekund (necelých 17 minut). Pokud jsou pakety generovány po 5 sekundách, tak minimální počet paketů musí být $1000/5 = 200$ [2]. Pro počítače s nižší frekvencí TCP časových známek je pak interval a počet paketů vyšší. Pokud se navíc odchylka v čase výrazně mění, minimální počet paketů nebo délky měření není možné stanovit.

S ohledem na výledky experimentu byla stanovena minimální doba měření alespoň jedna hodina. Z 200 paketů nebo za 10 minut totiž program dokáže vypočítat pouze odchylky, které se v čase nemění. Intervaly mezi pakety se pohybovaly od 1 do 5 sekund podle počtu měřených zařízení tak, aby program nebyl zbytečně zatížen velkým množstvím zpracovávaných paketů.

5.2 Experimenty zaměřené na software

V původním článku [5], který přinesl myšlenku identifikace počítače na základě odchylky hodin, byly jako možné metody získání časové známky uvedeny a otestovány TCP známky a ICMP zprávy. Oba tyto přístupy mají určité nevýhody v porovnání s časovými známkami Javascriptu. Javascript není blokován firewally po cestě, nemusí se upravovat hodnoty v registrech a je ve většině počítačů povolen. Na rozdíl od ostatních přístupů ale generuje známky na aplikační úrovni. V této sekci bylo provedeno několik typů experimentů. Každý z nich byl zaměřen na porovnání výsledků získaných z různých zdrojů časových známek a vyloučení či potvrzení vlivu softwarové konfigurace na odchylku hodin. U některých operačních systémů se vyskytly komplikace popsané v sekci 5.1.2 a nebylo možné využít všechny způsoby měření. Přehled měřených operačních systémů a vypočítaných frekvencí je uveden v tabulce 5.3.

Operační systém	Frekvence [Hz]
Windows XP	10
Windows 7, 8	100
Android 4.0.4	100
Ubuntu 12.04	250
FreeBSD 9.0	1000
Mac OS X 10.8	1000

Tabulka 5.3: Frekvence měřených operačních systémů.

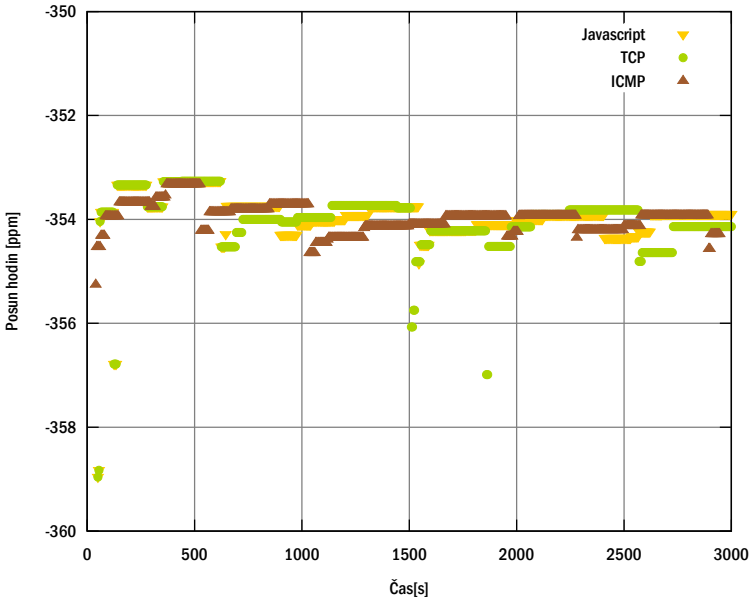
5.2.1 Porovnání metod pro získání časových značek

Cílem prvního experimentu je zjištění rozdílů mezi odchylkami vypočítanými z TCP, ICMP a Javascriptu. Jak již bylo uvedeno v práci [4], operační systém ovlivňuje frekvenci časových známek TCP. Javascript má stejně jako ICMP pevně danou frekvenci 1 kHz nezávisle na operačním systému. Testy byly rozvrženy do kategorií podle operačního systému, kde se rozdíly mezi vypočítanými hodnotami projeví na první pohled.

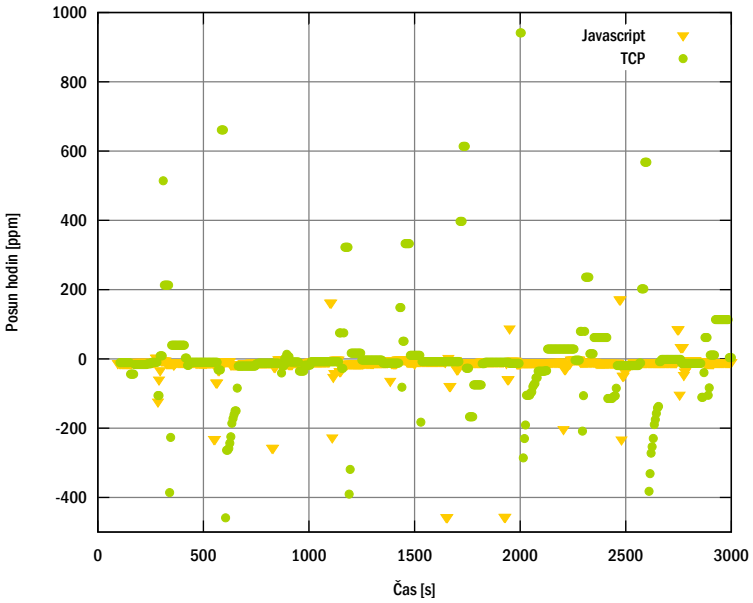
Pro první typ experimentu byl posun hodin vypočítán po každém přijatém paketu s intervalem 5 sekund. V grafech 5.3 a 5.4 je znázorněno, jak se od sebe jednotlivé metody liší. V případě operačního systému Ubuntu 12.04 (graf 5.3) je rozptyl mezi TCP, ICMP a Javascriptovými výsledky velmi malý. Výsledný posun vypočtený pomocí TCP odpovídal hodnotě -354,050 ppm, ICMP -353,918 ppm a Javascript -354,009 ppm. V průběhu výpočtu se posun hodin krátkodobě lišil až o 7 ppm, což bylo pravděpodobně způsobeno právě vypočítáváním odchylky po každém přijatém paketu.

U Windows 8 (graf 5.4) se jednotlivé metody lišily více. V první řadě mělo měření pomocí Javascriptu vyrovnanější hodnoty než pomocí TCP. Javascript se v průběhu měření pohyboval od -14 do -6 ppm (rozdíl 8 ppm) a občasné výkyvy se od těchto hodnot lišily maximálně o 20 ppm. Posun hodin vypočítaný z TCP se pohyboval v rozmezí -20 až 28 ppm (rozdíl 48 ppm) s občasnými výkyvy až o 300 ppm. Windows 8 je tedy z pohledu TCP systém, kde se odchylka v čase poměrně často mění. Další měřené operační systémy Windows XP a Windows 7 vykazovaly podobné chování (grafy jsou přiloženy na DVD). Mac OS X, který vykazoval jiné nezvyklé chování je popsán v sekci 5.2.3. Javascript u obou těchto systémů kolísal méně a dlouhodobé měření po blocích paketů tuto domněnku potvrdilo.

Druhý typ experimentu porovnával TCP a Javascript v dlouhodobém měření. Od prvního se lišil především zapojením počítačů ze strany uživatelů Internetu a tedy získáním reálných dat. Pro tento typ měření byl výpočet odchylky pozastaven a výstupem programu byly pouze logy, které se zpracovaly pomocí nástrojů popsaných v sekci 4.2. Výsledky veřejného měření jsou uvedeny v příloze na DVD. Mezi TCP a Javascriptem je jasně zřejmá vysoká míra korelace. IP adresy, u kterých jsou si odchylky nejvíce podobné, pochází ze stejné zdrojové IP adresy. Až na některé výjimky, které měly vyšší rozdíl posunů, je zřejmé, že TCP a Javascript jsou si velmi podobné a liší se maximálně o 1 ppm. Zároveň je ale velmi pravděpodobné, že se nejednalo o počítače s operačním systémem Windows (nebyla změněna hodnota v registru, která by umožnila odesílání TCP razítek) nebo Mac OS X (TCP nelze provnávat s Javascriptem).



Obrázek 5.3: Vliv použité metody na posun hodin u Ubuntu 12.04.



Obrázek 5.4: Vliv použité metody na posun hodin u Windows 8.

5.2.2 Prohlížeče

Tento experiment je zaměřen na porovnání posunů hodin v závislosti na použitém prohlížeči. Časové známky byly generovány pouze pomocí Javascriptu, protože na TCP nemá použitý prohlížeč žádný vliv. Časová známka se vytváří na aplikační úrovni pomocí funkce `getTime` a je odeslána na webový server v GET požadavku. Funkce `getTime` je podporována ve všech používaných prohlížečích a ve všech případech byla zjištěna frekvence 1 kHz. Výsledky měření jsou uvedeny v tabulce 5.8. U všech systémů se posuny hodin pohybovaly v rozmezí +- 0,6 ppm. Použitý prohlížeč tedy vliv na odchylku a tedy i posun hodin nemá.

Prohlížeč	Windows 7	Windows 8	Ubuntu 12.04
Mozilla Firefox	-13,041	-12,372	-353,875
Chrome (Chromium)	-13,329	-12,158	-354,215
Internet Explorer	-13,509	-13,516	–
Midori	–	–	-354,023

Tabulka 5.4: Vliv použitého prohlížeče na posun hodin (v ppm).

5.2.3 Mac OS X

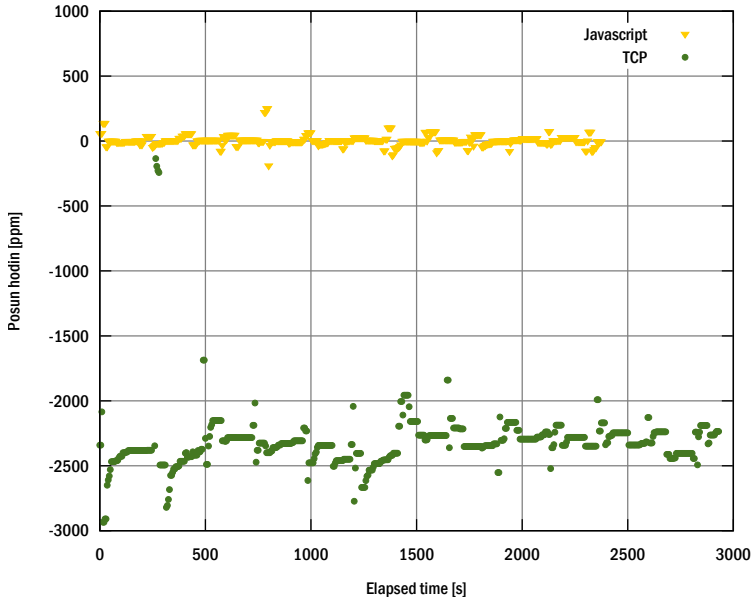
Samostatnou kapitolou jsou produkty firmy Apple. Předběžné výsledky experimentů naznačovaly nestandardní chování odchylek vypočítaných z TCP časových známek. V původní práci [5] ani v žádné další nebylo zmíněno, že by se tato zařízení chovala jinak než ostatní. TCP ovšem při měření vykazovalo posun hodin v řádech tisíců ppm na rozdíl od všech ostatních zařízení, které 1000 ppm ve většině nepřekročily.

Experimenty byly provedeny na třech zařízeních v laboratoři: Macbook Air, Mac Mini a IPad. Všechny potvrdily nezvykle vysoký posun hodin získaný z TCP. Zároveň jde při použití metody TCP o extrémně nestabilní systém, což jde vidět v grafech 5.5 i 5.6. Důvody změn v odchylce a zřejmých skoků se nepodařilo diagnostikovat. Během některých měření měla odchylka podobu přímky, která ale byla i tak nestabilní a hodnoty se měnily v řádu stovek ppm. Pro porovnání byl proveden test s virtualizovaným systémem Mac OS X a byl vyloučen vliv hardware. Stejně tak byla provedena řada experimentů se zatížením procesoru, které ovšem nepřinesly očekávané výsledky. Vliv zatížení procesoru se nepotvrdil a s tím související teplota baterie také ne. Dalším testovaným případem byla instalace aktualizací a běžné používání zařízení během měření, které také neovlivnily odchylky stylem, který lze pozorovat v grafu 5.6.

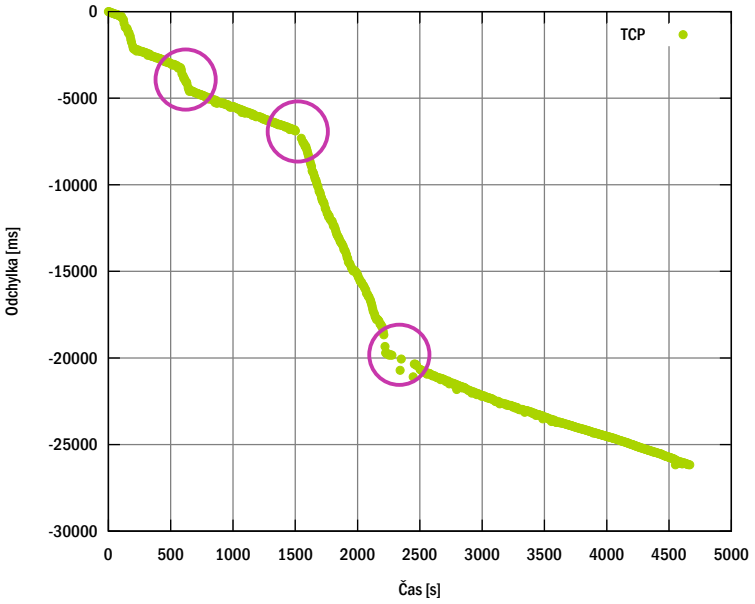
Odchylka hodin byla v některých článcích prováděna pomocí ICMP, ale všechny použité verze operačního systému tuto variantu nepodporovaly (viz. tabulka 5.1). Pro porovnání je v grafu 5.7 uvedena odchylka, která byla vypočítána z Javascriptu. Trend odchylky hodin získané z Javascriptu odpovídá ostatním naměřeným zařízením a nesouhlasí s odchylkou získanou z TCP. Výsledný vypočítaný posun byl -0,888 ppm. Je to hodnota poměrně vysoká, ale nerozdíl od TCP se hodnoty pohybují o tři řády níž.

5.2.4 Vliv NTP

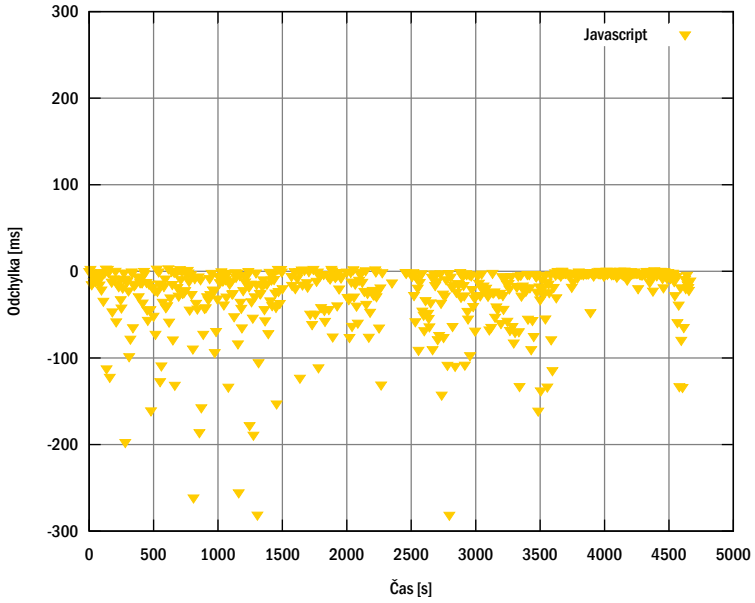
Experimenty provedené v [4] ukazují značný rozdíl vlivu synchronizace na časové známky TCP u různých operačních systémů. Dle již provedených testů operační systém Linux se



Obrázek 5.5: Rozdíl posunu hodin mezi jednotlivými metodami u počítače Mac Mini.



Obrázek 5.6: Odchylka hodin počítače Mac Mini měřená metodou TCP. Ve vyznačených místech došlo z neznámého důvodu k výrazné změně trendu.



Obrázek 5.7: Odchylka hodin počítače Mac Mini měřená metodou Javascript.

spuštěným démonem nevykazuje prakticky žádný posun vnitřních hodin. Pokud se ale synchronizace provede pouze jednorázově, je možné určit odchylku. Naopak u FreeBSD by synchronizace neměla mít na TCP známky žádný vliv. U systémů Windows je automaticky nastavena synchronizace hodin jedenkrát týdně. Výsledky měření na mobilních telefonech naznačovaly, že odchylka by mohla být stabilní i s běžícím synchronizačním démonem [11].

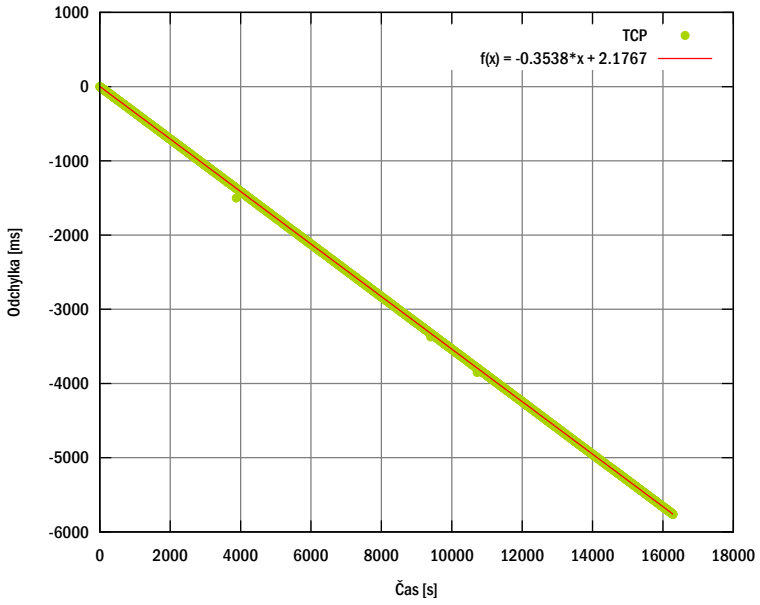
Experimenty s operačním systémem Windows 8 potvrdily předpoklad, že ani jeden typ časových značek není ovlivněny synchronizací hodin. Chytré telefony s operačním systémem Android 4.0.4 nebylo možné otestovat, protože standardní verze systému neumožňuje úpravu systémových hodin.

V [10] byly provedeny experimenty, které ověřily vliv NTP na TCP známky u operačního systému Linux. Výsledkem bylo zjištění, že u verze jádra 2.6.20 časové známky ovlivněny nejsou, ale od verze 2.6.22 vydané v roce 2007 se synchronizace hodin pomocí nástrojů *ntpd* i *ntpdate* projeví.

V rámci provedených experimentů v [4] byl potvrzen velký vliv synchronizace času na odchylku změřenou pomocí TCP časových známek. Cílem tohoto experimentu je prokázání vlivu NTP na časové známky získané z ICMP a Javascriptu. Samotný test byl proveden na operačních systémech Linux 12.04 (verze jádra 3.2.0) a FreeBSD 9.0 v laboratoři za pomoci programu ntpdate a ntpd. Ntpdate funguje jako jednorázová synchronizace, která podle velikosti odchylky hodin upraví čas podle vzdáleného serveru. Pokud má počítač odchylku menší než 5 ms/s, je tato změna provedena voláním funkce `adjtime`. Pokud je ale odchylka hodin větší, ve výchozím nastavení program posune hodiny funkcí `settimeofday`.

Experimenty byly provedeny s využitím časových známek ze všech tří zdrojů (TCP, ICMP, Javascript). Některé grafy si byly velmi podobné a v takovém případě je vždy uveden pouze jeden z nich jako ilustrace. Kolekce všech grafů je součástí přiloženého DVD.

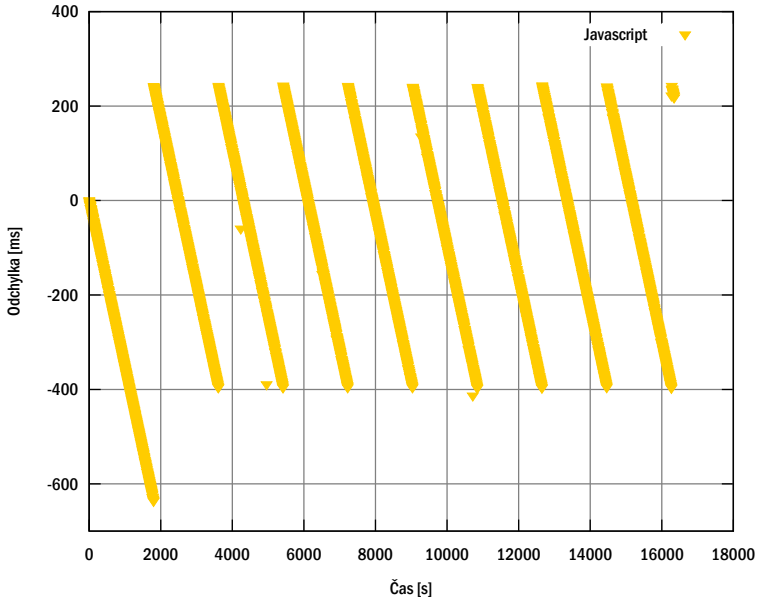
V první fázi experimentu byla změřena odchylka bez jakékoliv synchronizace hodin. Ve chvíli, kdy byla získána stabilní hodnota -354 ppm (kolísání hodnoty menší než 1 ppm), byla zapnuta synchronizace času pomocí ntpdate. V grafech 5.8 a 5.9 jsou znázorněny změny odchylky způsobené funkcí `settimeofday` u operačního systému Ubuntu 12.04. Časové známky získané z TCP nejsou tímto způsoben synchronizace ovlivněny vůbec. Vzhledem k tomu, že program PCF je schopný vypočítat i měnící se odchylku, a pokud vezmeme jako výsledek pouze posun hodin -354 ppm, měla by tato hodnota u ICMP a Javscriptu zůstat nezměněna i po synchronizaci.



Obrázek 5.8: Vliv synchronizace času pomocí funkce *settimeofday* na odchylku hodin u TCP.

V případě, že se odchylka nastavuje za pomoci postupného upravování systémových hodin, změna je patrná i ve vypočítaných hodnotách. Čím je odchylka systémových hodin počítače a serveru větší, tím déle je ovlivněna hodnota výsledku. V grafu 5.10 byla synchronizace prováděna každou hodinu a systémový čas se od reálného lišil o více než jednu sekundu. Vyrovnání tak velkého zpoždění hodin trvalo programu více než 30 minut, během kterých se odchylka -354 ppm změnila na 145 ppm. Po úspěšné synchronizaci času se odchylka opět vrátila na původní hodnotu.

Druhý experiment byl proveden za pomoci NTP démona. Synchronizace hodin se spustila po deseti minutách měření, kdy byl získán stabilní posun hodin -354 ppm. Démon `ntpd` ovlivňuje všechny typy generování časových známek, jak lze vidět v grafech 5.11 a 5.12. Po dvouhodinovém měření se posun postupně dostane až na hodnotu 0,2 ppm. Následným vypnutím démona se hodiny nevrátí zpět k původní odchylce, protože v souboru `/var/lib/ntp/ntp.drift` se spuštěním ntpd vytvoří konstanta, podle které se upraví hodiny při každém spuštění systému. Pokud synchronizační démon běžel dostatečně dlouho, i po jeho vypnutí se posuny hodin pohybují v rozmezí hodnot nižších než 0,1 ppm. V grafu 5.12 je navíc patrný velký skok ve chvíli, kdy *ntpd* začne ovlivňovat odchylku hodin.



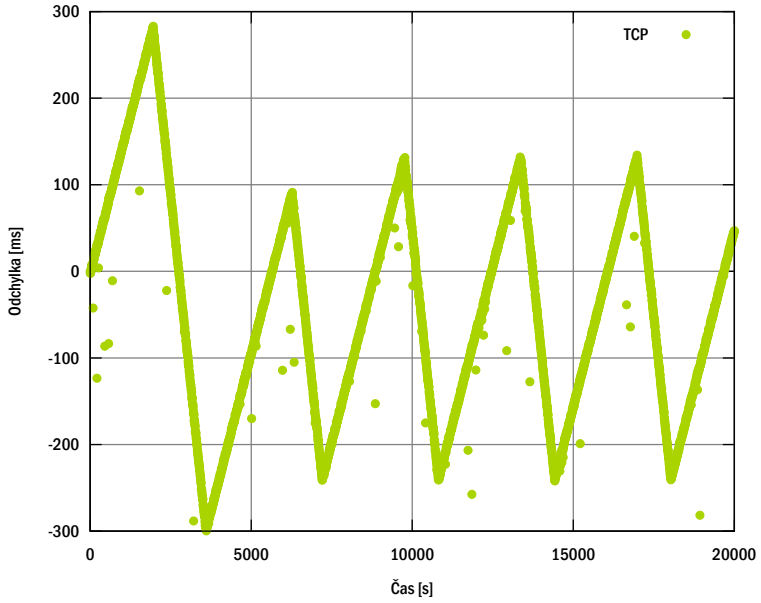
Obrázek 5.9: Vliv synchronizace času pomocí funkce *settimeofday* na odchylku hodin u Javascriptu (ICMP graf je analogický).

5.2.5 Párování IPv4 a IPv6

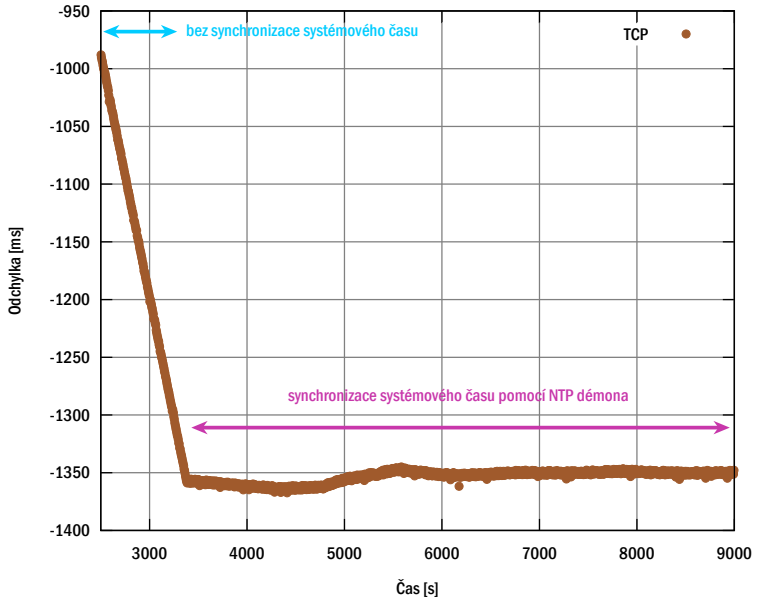
Odchylku hodin lze kromě identifikace zařízení použít i k párování IP adres v reálném čase. V tomto případě je hlavním problémem okamžité přiřazení dvou adres k jednomu počítači. Lze využít jak TCP časové známky, protože IPv4 i IPv6 sdílejí transportní vrstvu, tak Javascript. Samotný program porovnává všechny adresy na shodu, ale využívá k tomu pouze informaci o poslední potvrzené odchylce. Pokud tedy pár adres bude mít větší rozdíl než je 1 ppm, program ho nedokáže odhalit. Tato metoda je vhodná pro operační systémy se stabilní odchylkou, které v průběhu měření získají pouze jednu rovnici odchylky. Vliv má také výběr metody, pomocí které získáváme časové známky. Porovnání Javascriptu a TCP je uvedeno v sekci 5.2.1 a obecně jsou výsledky naměřené Javascriptem stabilnější. V ideálním případě stabilního systému a vhodné metody je pravděpodobnost úspěšného nalezení páru poměrně vysoká.

Analytické nástroje, popsané sekci 4.2, byly vytvořeny za účelem komplexního porovnání IP adres na shodu a vypočítají vážený průměr rozdílů odchylek. U každé dvojice je určeno o kolik ppm se od sebe liší, což nám umožní relativní pohled na porovnání. K dané IP adrese tedy budeme vždy hledat co nejmenší rozdíl odchylek, který je s největší pravděpodobností druhý do páru.

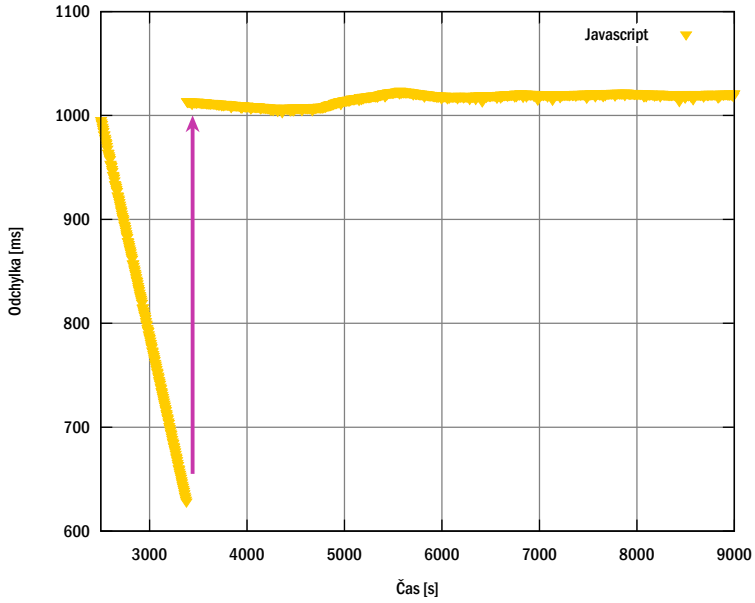
Samotné měření bylo rozděleno do dvou fází. V první fázi byly provedeny na známých zařízeních v laboratoři pomocí časových známek z Javascriptu. Tím byla umožněna zpětná kontrola výsledků, zda spárované IP adresy opravdu sedí nebo ne. Výsledky měření jsou uvedeny v tabulce 5.5. Z šesti měřených zařízení se podařilo všechny úspěšně spárovat. Maximální rozdíl posunů hodin mezi IPv4 a IPv6 u páru byl 2,18 ppm u Windows 8. Ten se v programu PCF spárovat nepodařilo, protože práh nastavený na odlišení dvou IP adres je 1



Obrázek 5.10: Vliv synchronizace času pomocí funkce *adjtime* na odchylku hodin u TCP (Javascript a ICMP grafy jsou analogické).



Obrázek 5.11: Vliv synchronizace času pomocí démona *ntpd* na odchylku hodin u TCP.



Obrázek 5.12: Vliv synchronizace času pomocí démona *ntpd* na odchylku hodin u Javascriptu (ICMP graf je analogický).

ppm. Pokud ale porovnáme hodnotu relativně vzhledem k ostatním vypočítaným rozdílům, nejbližší vyšší hodnota je 3,54 ppm, což znamená, že ačkoliv je rozdíl tak velký, lze s vysokou pravděpodobností určit pár s rozdílem 2,18 ppm. Tento závěr ovšem můžeme udělat pouze u měření s omezeným počtem zařízení v laboratoři, kde lze spárovat IPv4 a IPv6 adresy v poměru 1:1. Pro využití v praxi je vypočítaný rozdíl moc vysoký a zařízení by nemuselo být možné spárovat (viz tabulka veřejného měření 5.7).

IPv4 x IPv6	**Apple 1**	**Apple 2**	**Apple 3**	**Ubuntu 1**	**Ubuntu 2**	**Windows 8**
Apple 1	**0,67**	1,76	3,17	2,38	350,39	5,96
Apple 2	1,3	**0,21**	5,06	1,16	352,36	8
Apple 3	3,71	4,79	**0,05**	5,39	347,35	2,99
Ubuntu 1	2,01	0,92	5,77	**0,35**	353,07	8,71
Ubuntu 2	351,08	352,17	347,32	352,65	**0,03**	344,39
Windows 8	6,27	7,36	3,54	8,72	344,44	**2,18**

Tabulka 5.5: Rozdíl IPv4 a IPv6 zařízení v laboratoři měřených pomocí Javascriptu (v ppm).

U operačního systému Mac OS X měřeného metodou TCP ale výsledky nevycházely tak přesně jako u Javascriptu. Vypočítané rozdíly posunů jsou uvedeny v tabulce 5.6. Zařízení Apple 1 se neúspěšně spárovalo se zařízením Apple 2 a rozdíl byl 12,79 ppm. Rozdíl mezi IPv4 a IPv6 u Apple 1 pak byl 22,29 ppm. Další dvě zařízení se úspěšně spárovala. Hodnoty rozdílů jsou mnohem vyšší než v případě Javascriptu, ale jak je uvedeno v sekci 5.2.3,

odchylka hodin u operačního systému Mac OS X může být až 3000 ppm. Tento trend jasně odlišuje operační systém Mac OS X od ostatních, což sníží pravděpodobnost spárování jakéhokoliv zařízení Apple s jiným operačním systémem.

IPv4 x IPv6	**Apple 1**	**Apple 2**	**Apple 3**
Apple 1	22,29	**12,5**	900,76
Apple 2	17,48	**7,69**	895,95
Apple 3	870,84	880,63	**7,63**

Tabulka 5.6: Rozdíl IPv4 a IPv6 zařízení Apple měřených metodou TCP (v ppm).

Druhá fáze testování proběhla veřejně. Nebylo možné zajistit minimální délku měření a zjistit, které IP adresy skutečně tvoří pár. Využity byly časové známky získané pomocí Javascriptu, kterých bylo mezi získanými daty více než známek z TCP. V tabulce 5.7 jsou uvedeny vybrané výsledky. Celá tabulka s porovnáním IP adres je v příloze na DVD.

Celkově bylo analyzováno čtrnáct IPv6 adres proti padesátidevíti IPv4 adresám. S vysokou pravděpodobností, že se opravdu jedná o odpovídající pár, se podařilo vytvořit pouze dva páry a to u počítačů *host 1* a *host 2*. Rozdíly odpovídajících párů byly 0,24 ppm a 0,21 ppm. V rámci celé tabulky už nebyla pro tyto IPv6 adresy nalezena další shoda. Je nutné zmínit, že na počítačích *host 10* až *host 14* pravděpodobně běžel synchronizační démon, protože rozdíly odchylek se pohybovaly v rozsahu 0,02 až 1 ppm a byly jim často přiřazeny stejné IPv4 adresy. Získat z těchto dat skutečný pár není možné ani jednou použitou metodou. Další zařízení se nepodařilo identifikovat. V tabulce jsou uvedeny počty IPv4 adres, které by těmto IPv6 adresám přicházely v úvahu na přiřazení do dvojice.

IPv6 adresy	**nejnižší rozdíl**	**nejbližší další rozdíl**	**počet možných IPv4 párů**
host 1	0,24	0,74	1
host 2	0,21	0,98	1
host 3	0,01	0,09	4
host 4	12,35	14,14	6-8
host 5	14,62	15,02	7-8
host 6	131,40	211,65	nelze určit
host 7	74,94	171,45	nelze určit
host 8	8,82	21,22	4 (do 30 ppm)
host 9	85,72	94,43	nelze určit
host 10	0,03	0,06	11 (do 1 ppm)
host 11	0,03	0,04	15 (do 1 ppm)
host 12	0,05	0,10	17 (do 1 ppm)
host 13	0,03	0,04	13 (do 1 ppm)
host 14	0,17	0,22	12 (do 1 ppm)

Tabulka 5.7: Analýza veřejného párování IPv4 a IPv6 (první a druhý sloupec v ppm).

5.3 Experimenty zaměřené na hardware

Vliv hardware a okolního prostředí byl zmíněn už v původní práci [5]. Systémové hodiny jsou generovány z hardwarových, což je ve skutečnosti mechanický oscilátor, proto je ovlivnění odchylky teplotou nebo zatížením přístroje velmi pravděpodobné. V této sekci jsou popsány experimenty, které se snaží prokázat, jak do jaké míry je odchylka ovlivněna teplotou, typem napájení, vzdáleností od měřícího zařízení a typem připojení k počítačové síti.

5.3.1 Teplota CPU

Frekvence krystalového oscilátoru je ovlivněna jeho teplotou. Míra ovlivnění je dána typem krystalu a technologií výroby [7]. V [4] byly provedeny testy, které se snažily prokázat vliv teploty procesoru na odchylku hodin. Teplotní rozdíl procesoru se pohyboval v závislosti na vytížení počítače od 38°C do 75°C. V tomto rozmezí se nepodařilo prokázat, že by teplota procesoru ovlivňovala vypočítanou odchylku hodin. Z jiného úhlu pohledu se danou problematikou zabýval J. Murdoch v [7], kde zkoumá vliv teploty procesoru na vypočítané odchylky a pomocí této metody detekovat útok na Tor. Tyto výsledky naopak vliv teploty potvrzují.

Cílem experimentu bylo prokázat vliv teploty na odchylku hodin. K docílení co největších rozdílů byla odchylka hodin měřena na počítačích s vytížením procesoru maximálně 2 %, které byly postupně zatíženy na 25, 50 a 100 % svého výkonu. Poté bylo zatížení opět sníženo na 2 %, aby se procesor mohl ochladit. I při opakovaném provádění experimentu však nebylo možné jednoznačně prokázat vliv teploty procesoru na odchylku hodin.

5.3.2 Baterie a napájení ze sítě

Tento experiment se zaměříme na dva stavy sledovaných zařízení. Cílem testů je prokázání zda má napájení ze sítě nebo provoz na baterii vliv na odchylku vnitřních hodin. V rámci experimentů provedených v [11] bylo zjištěno, že u mobilních telefonů a tabletů, které mají omezené výpočetní prostředky (CPU, paměť apod.) při provozu na baterii, je při nabíjení optimalizován výkon a tím jsou ovlivněny i odchylky hodin. Výsledky měření [4] naopak nezaznamenaly žádný rozdíl odchylky vnitřních hodin, který by byl způsoben aktuálním napájením nebo stavem baterie. V článku [5] je z grafu jasně patrný moment, kdy byl vypojen napájecí kabel i vliv nabíjení baterie při zapojeném napájecím kabelu.

Experimenty v rámci bakalářské práce byly provedeny na několika zařízeních včetně tabletu IPad a chytrých mobilních telefonů (např. Xperia Arc S). Během měření byla zařízení nejdříve zapojena do napájení a po získání stabilní odchylky bylo napájení vypojeno. V daném časovém okamžiku se odchylka hodin nezměnila a stejně tak tomu bylo i při opětovném zapojení napájecího kabelu. Vliv napájení tedy na odchylku také nebyl prokázán. Stejně tak rozdíl mezi odchylkou, která byla spočítána po nastartování systému se zapojeným napájením se nelišila od odchylky získané po nastartování systému pouze s baterií.

5.3.3 Umístění/vzdálenost (VPN)

V rámci tohoto experimentu se pokusíme zhodnotit, jak je odchylka vnitřních hodin ovlivněna vzdáleností měřeného počítače od serveru. Nebyla bohužel možnost vyzkoušet připojení z různých zeměpisných lokací, nicméně délku cesty jsme schopni ovlivnit využitím Virtual Private Network (VPN). VPN se využívá pro propojení dvou bodů pomocí soukromé nebo veřejné sítě. Klient se virtuálně připojí k virtuálnímu portu na serveru VPN.

Server ověří klienta a následně přenáší data mezi privátní sítí a klientem VPN, což zvýší počet hopů a RTT.

K docílení co nejvyšších rozdílů v délce cesty byly VPN servery vybrány v různých zeměpisných lokacích. Získání časových známek pomocí ICMP bylo opět nevhodné, protože získat odpověď na požadavek z WAN není reálné. Z naměřených výsledků pomocí TCP a Javascriptu vyplývá, že počet hopů a vzdálenost, kterou musí pakety s časovou známkou překonat, nemají na vypočítanou odchylku vliv. U operačního systému Windows 8 i Linux se vypočítané hodnoty sice liší, ale nejnižší a nejvyšší hodnota spadají do intervalu +- 1 ppm od výsledku nejkratšího měření.

OS	metoda	LAN	17 hopů	18 hopů	21 hopů
Windows 8	TCP	-10,209	-10,227	-9,499	-10,227
	Javascript	-10,199	-10,071	-11,228	-9,807
Ubuntu 12.04	TCP	-351,367	-351,853	-352,001	-351,126
	Javascript	-353,111	-352,454	-352,960	-353,109

Tabulka 5.8: Závislost posunu hodin na vzdálenosti (v ppm).

5.3.4 Typ připojení k počítačové síti

Poslední experiment se týkal rozdílných způsobů připojení k počítačové síti. Experiment byl proveden se zařízeními, které podporují alespoň dva různé typy připojení. Testy proběhly s notebooky, u kterých se dalo připojit do sítě pomocí Ethernetu a Wi-Fi a s mobilními telefony, připojenými přes telefonního operátora a Wi-Fi. Výsledky v [11] vykazovaly na velmi malý rozdíl odchylky hodin v závislosti na typu připojení (okolo 0,1 ppm).

V rámci měření jednoho zařízení skok o 0,1 ppm sice jsme schopni detekovat, ale vzhledem k tomu, že naměřené hodnoty u přístrojů mohou kolísat až o 1,2 ppm, skok o desetinu ppm nemůžeme jednoznačně prohlásit za změnu odchylky z důvodu přepnutí typu připojení k počítačové síti. Pro identifikaci počítače tak typ připojení k síti nemá vliv na odchylku hodin. Provedené experimenty opravdu nevykázaly výraznou změnu v odchylce, čímž potvrdily předpoklad, že typ připojení k počítačové síti nemá vliv.

5.4 Shrnutí experimentů

Všechny provedené experimenty pokrývají celou škálu faktorů, které by teoreticky mohly ovlivňovat odchylku hodin. Během měření bylo zjištěno několik problémů jako je nemožnost získání ICMP razítek z operačních systémů Windows a Mac OS X. Z Windows lze získat TCP známky až po úpravě hodnot v registech počítače.

První experiment byl zaměřen na určení minimální možné délky měření. Bylo zjištěno, že u systémů, kde se odchylka v čase nemění, stačí k získání výsledku 8 minut měření. Pro všechny provedené experimenty byla ale doba měření nastavena na 1 hodinu, porotože jsme pracovali i se systémy, kde se odchylka v čase měnila o více než 1 ppm.

Po porovnání odchylek vypočítané z TCP, ICMP a Javascriptu jsme dospěli k závěru, že odpovídající si trojice výsledků pro jeden počítač je velmi podobná. Rozšířené veřejné testy tento závěr potvrdily, protože se většinu TCP a Javascriptových výsldků podařilo

spárovat (tzn. výsledky z jedné IP adresy získané rozdílnými metodami měly rozdíl posunů hodin o méně než 1 ppm). Nejlépe využitelnou metodou je pak získávání časových razítek pomocí Javascriptu, který jako jediný není blokován firewally a je podporován ve většině používaných prohlížečů i ve výchozím nastavení.

Operační systém Mac OS X, který byl testován v rámci dalších experimentů, vykázal nezvyklé chování, které není zmíněno v žádném z použitých zdrojů literatury. Posun hodin získaný pomocí TCP se pohyboval v řádech tisíců ppm a sklon odchylky často se měnil. Při měření Javascriptem pak výsledky vycházely ve standardním rozmezí, což znemožnilo vytvoření páru u jedné IP adresy.

Vliv NTP na odchylku hodin u operačního systému Linux byl již zkoumán. Práce [10] tento fakt potvrzuje. Provedené experimenty ale odhalily rozdílný vliv synchronizace podle toho, jakým způsobem je čas aktualizován. V případě, že se provede jednorázový skok voláním funkce *settimeofday*, TCP časové značky vůbec nejsou ovlivněny. ICMP a Javscript ano, ale program umí počítat i měnící se odchylku hodin, takže vypočítaný posun by měl odpovídat TCP. Úprava systémových hodin funkcí *adjtime*, která posouvá systémový čas postupně, se naopak projevila velmi výrazně. U systémů, které mají posun hodin větší než 500 ms/s, se čas aktualizuje pomocí funkce *adjtime* velmi dlouho, což se stejně výrazně projeví i na TCP, ICMP i Javscriptu. Získáme většinou tedy dva výsledky, kdy jeden značí skutečný posun hodin a druhý posun hodin během jejich úpravy.

Nově provedenými experimenty bylo také párování IPv4 a IPv6 adres. Podle rozdílu posunů jsme se snažili najít takový pár, který by mohl být jednoho sledovaného zařízení. V laboratorních testech se podařilo všechny použité zařízení spárovat, ale s reálnými daty byla úspěšnost této metody velmi nízká. Porovnání bylo provedeno komplexnější metodou, která brala v úvahu nejen poslední vypočítaný posun, ale i dříve získané hodnoty spolu s jejich váhou (intervalem, na kterém platily).

Dále provedené experimenty spokusily ověřit vliv hardware na odchylku hodin sledovaných zařízení. V těchto testech bylo nejvíce rozporů s literaturou. Zvýšení teploty CPU podle [5] i [7] mělo prokazatelně dopad na naměřenou odchylku hodin. Testování v rámci bakalářské práce však nedokázalo tyto výsledky prokázat. Vliv vzdálenosti zařízení byl nasimulován použitím VPN serverů, které se nacházely co nejdále od měřícího zařízení. I přes vysoký počet hopů, které tak informace o časové známce musely urazit, byl vypočítaný rozdíl posunů menší než 1 ppm. Stejně tak se nepodařilo prokázat, že by měl na odchylku njaký vliv typ použitého připojení do počítačové sítě.

Kapitola 6

Závěr

Tato bakalářská práce vychází z článku [5], kde byla poprvé představena metoda pro identifikaci hodin pomocí odchylky vnitřních hodin, a diplomové práce [4], ve které byl navržen a implementován program PCF. Program PCF zachytává příchozí pakety s časovým razítkem a s jejich pomocí určuje odchylku vnitřních hodin sledovaného počítače. Tento program byl v rámci bakalářské práce rozšířen tak, aby bylo možné spočítat odchylku hodin nejenom z TCP paketů, ale i z ICMP zpráv a z protokolu HTTP s využitím Javascriptu. Byly také navrženy a naimplementovány nástroje, které umožnily provést komplexnější analýzu výsledků na základě diskretizace a porovnání všech naměřených posunů hodin.

Pro program PCF byla navržena sada testů, které ověřily vliv software i hardware na výpočet ochylky hodin. Zajímavé výsledky přinesly zejména experimenty se synchronizací času pomocí *ntpdate*, kdy způsob synchronizace ovlivnil pouze některé metody sbírání časových známek. Jednorázová úprava hodin pomocí *settimeofdate* se u TCP neprojevila, ale u dalších dvou způsobů ano. Naopak úprava času funkcí *adjtime* se projevila ve všech metodách a způsobila zkreslení měření po dobu aktualizace času.

Nově provedeným experimentem bylo také komplexnější porovnávání IPv4 a IPv6 adres. Při experimentech v laboratoři dokázaly analytické nástroje úspěšně spárovat všechna zařízení měřená pomocí Javascriptu. U TCP pak došlo k jedné mýlce u operačního systému Mac OS X.

Porovnání různých metod sběru dat ukázalo na vysokou míru korelace mezi TCP a Javascriptem (kromě zařízení s operačním systémem Mac OS X). Při veřejném testování se podařilo většině IP adres spárovat TCP a Javascript s využitím implementovaných analytických nástrojů.

Možným navazujícím výzkumem by mohla být například komplexní analýza posunů hodin. Pouhým zahrnutím všech posunů vypočítaných v rámci aktuálního měření se zvýší pravděpodobnost úspěšného spárování dvou vstupních souborů. Pokud by byl posun specifikován například křivkou, mohli bychom získat více informací. Další možností je vytvoření sondy, která bude kromě časových razítek získávat v kombinaci s dalšími programy získávat více informací o sledovaných počítačích (operační systém, prohlížeč, vzdálenost). Data by mohla být v pravidelných intervalech odesílána k analýze, která by tak mohla vycházet z více dat a identifikace sledovaného zařízení by mohla být provedena s větší přesností.

Literatura

[1] Graham, R. L.: An efficient algorith for determining the convex hull of a finite planar set. *Information processing letters*, ročník 1, č. 4, 1972: s. 132–133.

[2] Huang, D.-J.; Yang, K.-T.; Ni, C.-C.; aj.: Clock Skew Based Client Device Identification in Cloud Environments. In *2012 IEEE 26th International Conference on Advanced Information Networking and Applications (AINA)*, 2012, s. 526–533.

[3] Jacobson, V.; Braden, R.; Borman, D.: TCP Extensions for High Performance [online]. `http://tools.ietf.org/html/rfc1323`.

[4] Jirásek, J.: Využití časových informací pro identifikaci počítače, diplomová práce, Brno, FIT VUT v Brně, 2012.

[5] Kohno, T.; Broido, A.; Claffy, K.: Remote Physical Device Fingerprinting. *IEEE Transactions on Dependable and Secure Computing*, ročník 2, č. 2, 2005: s. 93–108.

[6] Lanze, F.; Panchenko, A.; Braatz, B.; aj.: Clock Skew Based Remote Device Fingerprinting Demystified. In *Proceedings of the 55th International IEEE Global Communications Conference (IEEE GLOBECOM 2012)*, IEEE Computer Society Press, prosinec 2012, s. 831–837.

[7] Murdoch, S. J.: Hot or not: revealing hidden services by their clock skew. In *CCS 2006*, ACM Press, c2006, s. 27–36.

[8] Murdoch, S. J.; Lewis, S.: Embedding covert channels into TCP/IP. In *Information hiding*, Springer, c2005, s. 247–261.

[9] Novotný, J.: Identifikace počítače na základě časových značek paketů, bakalářská práce, Brno, FIT VUT v Brně, 2012.

[10] Polčák, L.; Jirásek, J.; Matoušek, P.: Comment on Remote Physical Device Fingerprinting. Submited to *IEEE Transactions on Dependable and Secure Computing*. 2012.

[11] Sharan, H.; Hussain, A.; Sharma, S.: Experience with heterogenous clock-skew based device fingerprinting. In *LASER '12 Proceedings of the 2012 Workshop on Learning from Authoritative Security Experiment Results*, ACM, 2012, s. 9–18.

[12] 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, 2008, s. 211–225.

Seznam příloh

Příloha A

Obsah DVD

app	kompletní balík software
tex	zdrojové soubory technické zprávy
bp-xfrank08.pdf	technická zpráva ve formátu PDF
bp-xfrank08-print.pdf	technická zpráva ve formátu PDF pro tisk
manual	nápověda k˜obsahu DVD
public-tests.xlsx	analýza veřejných testů ve formátu MS Excel
graphs	původní výstupy programu PCF